Dynamische Bewertung von Ressourcenbelegungsplänen durch Simulation

Diplomarbeit im Fach Informatik
vorgelegt von

cand. inform.
Johannes Beck

Betreuer:

Prof. Dr. Frank Puppe
Dipl.-Inform. Christian Hestermann

Lehrstuhl für Künstliche Intelligenz
und Angewandte Informatik
Universität Würzburg

Ich erkläre, die vorliegende Diplomarbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben.

Würzburg, den 15. Mai 1998

Inhaltsverzeichnis

1. EINLEITUNG 7

1.1. MOTIVATION 7
1.2. PROBLEMSTELLUNG 7
1.3. IMPLEMENTATION 8
1.4. AUFBAU DER AUSARBEITUNG 8

2. SIMULATION ALS PROBLEMLÖSUNGSMETHODE 9

2.1. BEGRIFFSDEFINITIONEN 9
2.2. KLASSIFIKATION VON MODELLTYPEN 10
2.3. DISKRETE EREIGNISGESTEUERTE SIMULATION 12
2.4. ABLAUF EINER SIMULATIONSSTUDIE 14
2.5. ZUSAMMENFASSUNG 17

3. FORMALE METHODEN ZUR MODELLIERUNG 19

3.1. EVENT GRAPHS (EREIGNISGRAPHEN) 19
3.2. DISCRETE EVENT SYSTEM SPECIFICATION (DEVS) 22
3.3. PETRI-NETZE 25
3.4. WEITERE FORMALE METHODEN 31
3.5. MISCHFORMEN 37
3.6. ZUSAMMENFASSUNG 37

4. PRODUKTIONSPROZESS UND RESSOURCENBELEGUNGSPLÄNE 39

4.1. ÜBERSICHT 39
4.2. EINFÜHRUNG IN RESSOURCENBELEGUNGSPLÄNE 40
4.3. EINSATZGEBIETE FÜR SIMULATION IN PPS-SYSTEMEN 46
4.4. STATISCHE BEWERTUNGSKRITERIEN FÜR RESSOURCENBELEGUNGSPLÄNE 47
4.5. STOCHASTISCHE EIGENSCHAFTEN DES FERTIGUNGSPROZESSES 51
4.6. BEWERTUNGSKRITERIEN FÜR DYNAMISCHE EIGENSCHAFTEN VON BELEGUNGSPLÄNEN 53
4.7. ZUSAMMENFASSUNG 56

5. MODELLIERUNG EINER AUFTRAGSGESTEUERTEN FERTIGUNG IN SIMULATOREN 57

5.1. KLASSIFIKATION VON SIMULATIONSSOFTWARE 57
5.2. ARENA 57
5.3. SIMPLE++ 60
5.4. ROSI 62
5.5. ZUSAMMENFASSUNG 64

6. DYBBS 65

6.1. EINFÜHRUNG IN COKE UND WIZARD 65
6.2. BEGRIFFSDEFINITIONEN 67
6.3. ARCHITEKTUR UND IMPLEMENTIERUNG 68
6.4. ZUSAMMENFASSUNG 78

7. EVALUATION 79

7.1. M / M / 1 - WARTESCHLANGENMODELL 79
7.2. SCHEDULING IN DYBBS 80
7.3. MASCHINEN-PERSONAL-MODELL 82
7.4. EILAUFTRÄGE EINPLANEN 84
7.5. PARAMETERLAUF MIT SICHERHEITSPUFFERN 86
7.6. ZUSAMMENFASSUNG 88

8. ZUSAMMENFASSUNG, BEWERTUNG UND AUSBLICK 89

8.1. ZUSAMMENFASSUNG 89
8.2. IMPLEMENTATION 90
8.3. ERGEBNISSE 91
8.4. BEWERTUNG 92

8.5. AUSBLICK 93

9. VERZEICHNISSE 95

9.1. LITERATURVERZEICHNIS 95
9.2. ABBILDUNGSVERZEICHNIS 96
9.3. VERZEICHNIS DER WWW-ADRESSEN 97

ANHANG 99

A: MODELLBIBLIOTHEK 100
B: BENUTZERHANDBUCH 103
C: ZUFALLSZAHLENERZEUGUNG 120
D: SCHNITTSTELLENBESCHREIBUNG 123
E: DANKSAGUNG 126
F: CDROM 127

1. Einleitung

1.1. Motivation

Die Leistungsfähigkeit eines Unternehmens hängt in der heutigen Zeit zu einem großen Teil von einem effizienten innerbetrieblichen Ablauf ab. Gleichzeitig werden die Anforderungen der Kunden immer komplexer und spezifischer. Dies liegt u.a. in der zunehmenden Globalisierung der Märkte begründet, in denen die Unternehmen einem größeren Kunden- und Konkurrentenkreis gegenüberstehen.

Für eine effiziente Unternehmensführung sind Ablaufpläne notwendig, die zum einen auf die immer komplexeren Produktionsvorgänge eingehen und zum anderen mit den vorhandenen Ressourcen möglichst effizient umgehen. Zu diesen Ressourcen gehören u.a. Beschäftigte, Kapital, Produktionsmittel und Zeit. Aufgrund von Größe und Komplexität ist eine manuelle Erstellung eines guten Ablaufplanes nicht mehr möglich, weshalb seit mehreren Jahrzehnten Computersysteme zur Lösung dieses Problems verwendet werden. Produktionsplanungs- und -steuerungssysteme (PPS) werden heutzutage in verschiedenen Formen und Größenordnungen in fast allen großen und mittelständischen Unternehmen eingesetzt.

Allerdings kommt es häufig zu Problemen beim Einsatz von PPS-Systemen, wenn diese von den Beschäftigten nicht oder nur teilweise akzeptiert werden, wobei ein Hauptkritikpunkt in der mangelnden Übereinstimmung der Ablaufpläne mit der Wirklichkeit liegt. Dabei waren die Pläne zu ihrer Erstellungszeit sowohl korrekt als auch effizient. In der Zeit zwischen Planerstellung und Ausführung sind hingegen Ereignisse eingetreten, die in der Planung nicht berücksichtigt werden konnten, und die dazu führen, daß der Plan überholt ist. Treten überholte Belegungspläne im Fertigungsumfeld gehäuft auf, so führt dies zu einer generellen Ablehnung der durch das PPS-System erzeugten Belegungspläne. Die Folge ist eine Fertigungsumgebung mit einem unübersichtlichen und ineffizienten Produktionsprozeß, indem der Ablauf wie bisher von Schicht- und Abteilungsleitern lokal geplant wird, und der zusätzlich noch Kosten für das PPS-System verursacht.

1.2. Problemstellung

In einem Unternehmen unterliegen innerhalb der Produktion viele Prozesse zufälligen Schwankungen. Dazu zählen unter anderem die Dauer von Produktionsprozessen, Materialliefertermine und Ausschuß bei der Produktion. Dies hat zur Folge, daß für diese Größen nur Schätzwerte in die Planung eingehen können. Ebenso können Ereignisse auftreten, die generell nicht planbar sind, da ihr Auftreten zufallsabhängig ist. Zu diesen Ereignissen gehören z. B. die Störung einer Maschine oder der krankheitsbedingte Ausfall von Personal. Kein Planungssystem kann mit dem sicheren Eintreten solcher Ereignisse rechnen. Da diese Ereignisse aber häufig vorkommen, kann man Schätzungen über ihre Häufigkeit und Dauer angeben.

In PPS-Systemen werden diese zufälligen Prozesse typischerweise durch die Angabe von Sicherheitsgrößen behandelt, die zu den eigentlichen Bedarfen und Kapazitäten addiert werden. Im weiteren Verlauf werden alle Mengen und Termine innerhalb eines PPS-Systems deterministisch berechnet. So erstellte Belegungspläne können qualitativ und quantitativ nach den vier Grundzielen von PPS-Systemen – hohe Termintreue, geringe Lagerbestände, kurze Durchlaufzeiten, hohe Kapazitätsauslastung ([Ker94], S. 17) – bewertet werden. Zu geringe Sicherheitsaufschläge führen zu Belegungsplänen, die in der Realität nicht eingehalten werden können, zu hohe Aufschläge verteuern die Produktion, da sie die Auslastung verringern, Durchlaufzeiten verlängern und Lagerkosten erhöhen, garantieren aber eine hohe Termintreue. Die Angabe der Sicherheitsbestände wird in der Regel aus Erfahrungswerten und „Daumenregeln“ bestimmt. Inwieweit diese Schätzungen korrekt sind, läßt sich nicht direkt überprüfen. Die Robustheit des Belegungsplanes, d.h. die Fähigkeit des Planes, unter zufälligen Schwankungen und unvorhergesehenen Ereignissen seine Gültigkeit zu behalten, kann mit den vier Grundzielen nicht bewertet werden.

Aus diesem Grund benötigt der Disponent außer einem Werkzeug zur Planerstellung eine weitere Komponente zur Bewertung des Planes, in der zusätzliches Wissen über die nicht im Plan berücksichtigten stochastischen Eigenschaften des Produktionsprozesses eingeht. Eine Möglichkeit, eine solche Komponente zu verwirklichen, stellt die Modellierung und Simulation des Produktionsprozesses dar. Dabei wird zusätzlich zum Belegungsplan ein Modell des Produktionsablaufes benötigt, welches von einem Experten erstellt werden muß. Durch die Modellierung und Simulation des Produktionsprozesses mit seinen stochastischen Eigenschaften ist es möglich, Belegungspläne auf ihre Robustheit hin zu bewerten. Dadurch kann dem Disponenten ein Hilfsmittel gegeben werden, das es ihm erlaubt, Pläne zu erstellen, die auf der einen Seite ausreichend robust gegenüber unvorhergesehenen Ereignissen und auf der anderen Seite möglichst wirtschaftlich, d.h. mit möglichst geringen Sicherheitsreserven behaftet, sind.

1.3. Implementation

Im Rahmen dieser Arbeit wurde die Simulationsshell DYBBS (Dynamische Bewertung von Belegungsplänen durch Simulation) zur Simulation und Bewertung von Belegungsplänen implementiert. Die Implementation erfolgte in Common-LISP unter ALLEGRO COMMON LISP FOR WINDOWS[1] auf Basis des Betriebssystems MS WINDOWS 95/NT4.0. Die Implementierung ist vollständig in das PPS-Werkzeug WIZARD ([Hes97]) integriert.

1.4. Aufbau der Ausarbeitung

Für ein Simulationswerkzeug zur Unterstützung von PPS-Systemen werden Grundlagen über Simulation und PPS-Systeme benötigt. Diese Themengebiete werden zunächst getrennt untersucht. Zu Beginn wird in Kapitel 2 auf grundlegende Eigenschaften von Simulationen eingegangen. Es wird eine Klassifikation von Simulationsmodellen und ein Algorithmus, der die Simulation einer großen Klasse von Modelltypen ermöglicht, vorgestellt. Desweiteren wird ein Vorschlag zur Durchführung eines Simulationsprojektes erläutert. In Kapitel 3 werden verschiedene formale Methoden zur Erstellung eines Simulationsmodells vorgestellt. Drei dieser Methoden - Ereignisgraphen, DEVS und Petri-Netze - werden genauer betrachtet.

Die zur Simulation eines Fertigungssystems anhand eines Ressourcenbelegungsplanes notwendigen Grundlagen werden in Kapitel 4 umrissen. Dabei wird erläutert, welche Daten in PPS-Systemen abgelegt sind, und welche Ansätze zur Erstellung von Ressourcenbelegungsplänen existieren. Es werden Bewertungskriterien vorgestellt, die die Qualität des erstellten Belegungsplans prüfen. Insbesondere wird auf die Unterschiede zwischen statischen und dynamischen Bewertungen eingegangen.

Die in den beiden vorhergehenden Kapiteln erarbeiteten Grundlagen werden in den Kapiteln 5 und 6 zusammengeführt. Kapitel 5 bietet einen Überblick über drei Simulationswerkzeuge im Hinblick auf die gegebene Aufgabenstellung der Simulation eines auftragsgesteuerten Fertigungssystems mit Bewertung der eingehenden Belegungspläne. Dazu wurden zwei kommerzielle (ARENA und SIMPLE++) und ein akademisches (ROSI) Simulationswerkzeuge untersucht. Die im Rahmen dieser Arbeit implementierte Simulationsshell DYBBS wird in Kapitel 6 vorgestellt. Dabei wird auf die zugrundeliegenden Systeme COKE und WIZARD eingegangen, die Anforderungen an die Implementation vorgestellt und deren Realisierung beschrieben. Bei der Implementation gehen die in den Kapiteln 2, 3 und 5 vorgestellten Simulationsansätze in eine der Problemstellung angepasste Simulationsshell ein, die aufgrund ihrer engen Integration mit einem PPS-Werkzeug zur Untersuchung der in Kapitel 4 erläuterten Probleme geeignet ist.

In Kapitel 7 erfolgt anhand von fünf Beispielmodellen eine Evaluation von DYBBS. Dabei werden viele Einsatzmöglichkeiten der implementierten Simulationsshell anhand von kleinen und übersichtlichen Simulationsmodellen aufgezeigt. Kapitel 8 bietet eine Zusammenfassung und einen Ausblick an.

[1] Franz Inc.: http://www.franz.com/

2. Simulation als Problemlösungsmethode

Zu den wichtigsten Bestandteilen einer Simulation gehören das zu simulierende System und das Modell, das eine Abbildung eines Systems darstellt. Diese grundlegenden Bestandteile werden zu Beginn in Abschnitt 2.1 definiert. In Abschnitt 2.2 wird eine Klassifikation vorgestellt, nach welcher Simulation durch charakterische Eigenschaften des Modells eingeteilt werden kann. Der vielen Simulationsprogrammen zugrundeliegende Algorithmus der Diskreten Ereignisgesteuerten Simulation (DES) wird in Abschnitt 2.3 vorgestellt. Eine Vorgehensweise von der Analyse des Systems über die Bildung eines Modells und der Durchführung der Simulation bis hin zur Analyse der Ergebnisse, die sich an den Methoden des Softwareentwurfs orientiert, wird abschließend in Abschnitt 2.4 erläutert.

2.1. Begriffsdefinitionen

Um eine Basis für die weiteren Erläuterungen zu schaffen, sind zu Beginn einige einführende Definitionen notwendig. Eine für den Bereich der Informatik angemessene Definition einer ***Simulation*** stammt von Hoover und Perry:

> *„[Simulation is] the process of designing a mathematical or logical model of a real system and then conducting computer-based experiments with the model to describe, explain, and predict the behavior of the real system." ([Hoo89], S. 5)*
> *(Simulation ist der Prozess des Entwerfens eines mathematischen oder logischen Modells eines realen Systems und des Durchführens von computer-basierten Experimenten mit diesem Modell, um das Verhalten des realen Systems zu beschreiben, zu erklären und vorherzusagen.)*

Allgemeinere Definitionen schließen zusätzlich physikalische Modelle (z.B. der Nachbau eines Systems in kleinerem Maßstab), physikalische Experimente und imaginäre Systeme (z.B. geplante) mit ein. Der Begriff ***Simulationsstudie*** wird oft synonym zur obigen Definition verwendet. Simulation bezieht sich dann nur auf die Durchführung von Experimenten.

Die vorgestellte Definition macht deutlich, daß eine Simulation aus zwei Schritten besteht. Zuerst muß ein „System" „modelliert" werden und danach Experimente auf diesem Modell durchgeführt werden. Ein ***System*** ist dabei

> *„... a part of the world which we choose to regard as a whole, separated from the rest of the world for some period of consideration, a whole which we choose to consider as containing a collection of components, each characterized by a selected set of data items and patterns, and by actions which may involve itself [a component] and other components" ([Pag94], S.11)*
> *(... ein Bestandteil der Welt, den wir als ein Ganzes ansehen wollen, getrennt vom Rest der Welt für einen Betrachtungszeitraum, ein Ganzes, das wir als eine Sammlung von Komponenten auffassen wollen, die jede durch bestimmte Daten und Muster und durch Aktionen, die eine oder mehrere Komponenten umfassen, charakterisiert ist.)*

Diese Definition eines Systems macht deutlich, daß die Wahl, was als System und Komponenten angesehen wird, subjektiv von der Sichtweise und den Zielen des Modellierers abhängt. In einer mehr objektorientierten Sichtweise werden Komponenten auch als ***Objekte***, die Daten der Komponenten auch als ***Attribute*** oder ***Zustandsvariable*** und die Aktionen als ***Methoden*** bezeichnet. Die Entstehung objektorientierter Programmiersprachen wurde maßgeblich durch die Forschung auf dem Gebiet der Simulation beeinflußt. Die Simulationssprache SIMULA gilt als erste objektorientierte Programmiersprache.

Der ***Zustand einer Komponente*** ist eine Belegung aller Zustandsvariablen der Komponente mit einem Wert zu einem Zeitpunkt ([Pag94], S. 12). Der ***Zustand des Systems*** ist die Menge der Zustände der einzelnen Komponenten. Ein ***Modell*** ist eine Abstraktion des Systems, die dazu dient, einige Eigenschaften des Systems zu reproduzieren ([Pag94], S. 11). Im Abschnitt 2.2 wird eine Klassifikation von Modelltypen vorgestellt.

Simulation als abstraktes Modellieren eines Systems und Experimentieren am Modell ist dabei nur eine von mehreren Untersuchungsmethoden für das Verhalten von Systemen. [Hoo89] stellt folgende Untersuchungsmöglichkeiten gegenüber:

- **Experimentieren am realen System gegenüber Experimentieren am Modell:** wenn es möglich und billiger ist, Experimente am realen System durchzuführen, sollte diese Möglichkeit der Erstellung eines Modells vorgezogen werden, da keine Abstraktionen notwendig und die Ergebnisse gesicherter sind. Doch in den meisten Fällen kann nicht am System selbst experimentiert werden, da es noch nicht existiert, Experimente das System schädigen können oder zu teuer sind.
- **Physikalisches Modell gegenüber mathematisches Modell:** Physikalische Modelle bieten sich an, wenn eine Erstellung von abstrakten Modellen schwierig oder die Simulation eines abstrakten Modells sehr aufwendig wird. Häufig finden sich hier auch Mischformen, vor allem in Bereichen, die erst mit der Leistungskraft heutiger Rechner simulierbar werden, z.B. bei Windkanalversuchen oder Crashtests.
- **Analytische Lösung gegenüber Simulation:** ein abstraktes, mathematisches Modell kann zum einen simuliert werden, zum anderen kann man auch versuchen, eine analytische Lösung zu finden, z.B. durch das Lösen von Gleichungssystemen. Allerdings ist eine analytische Lösung nur für wenige Modelle möglich.

Die verschiedenen Möglichkeiten verdeutlichen, daß zu Beginn eine detaillierte Untersuchung des zu studierenden Systems und der genauen Festlegung der Ziele der Untersuchung erfolgen muß, bevor die Festlegung auf eine Methode und Modell getroffen werden kann. In Abschnitt 2.4 wird bei der Erläuterung der Durchführung einer Simulationsstudie noch genauer darauf eingegangen.

2.2. Klassifikation von Modelltypen

Ein Modell als Abstraktion eines realen Systems kann durch bestimmte Eigenschaften, die zur Klassifikation von Modelltypen verwendet werden können, charakterisiert werden. [Pag94] stellt folgende vier voneinander unabhängige Eigenschaften mit jeweils zwei Ausprägungen vor, die sowohl vom zu untersuchenden System als auch von den Absichten des Modellierers abhängen:

- **Modellrepräsentation:** *physikalisch oder abstrakt*
- **Zeit:** *statisch oder dynamisch*
- **Ziel der Studie:** *präskriptiv oder deskriptiv*
- **Lösungstechnik:** *analytisch oder numerisch*

Eine ***physikalische*** Modellrepräsentation ist eine Replik des zu untersuchenden Systems, oft in kleinerem Maßstab, die sich wie das Original verhält. Ein ***abstraktes*** Modell hingegen ist eine symbolische Beschreibung des Systems. Dies kann z.B. in logischen oder mathematischen Ausdrücken oder durch natürlichsprachliche Begriffe erfolgen. In einem ***statischen*** Modell sind die Modellparameter nicht von der Zeit abhängig, wohingegen in einem ***dynamischen*** Modell der zeitliche Verlauf eine Rolle spielt. Ein ***präskriptives*** Modell schreibt das Verhalten des Systems vor. Eine Lösung eines solchen Modells kann mit Bewertungen wie „optimal“, „suboptimal“, etc. versehen werden. Ein ***deskriptives*** Modell macht keine Aussagen über die Qualität des Verhaltens. Ein ***analytisches*** Modell kann durch formale Schlussfolgerungen gelöst werden, wobei sich eine exakte Lösung als Ergebnis ergibt. Ein ***numerisches*** Modell wird nur durch Berechnungen annäherungsweise gelöst.

Insgesamt ergeben sich somit durch Kombination 16 verschiedene Modelltypen, von denen viele auch sinnvoll sind. Beispielsweise kann ein Architekturmodell eines Hauses ein physikalisches, statisches und deskriptives und numerisches Modell sein, wenn es sich um einen Nachbau in kleinerem Maßstab handelt, bei dem die Statik des Modells über iterative Näherungsformeln bestimmt. Betrachtet man die Baupläne oder einen CAD-Entwurf anstatt eines Nachbaus, so wird das Modell abstrakt. Das Gleichungssystem, das die Bewegungen der Planeten um die Sonne bestimmt, ist abstrakt, dynamisch, deskriptiv und numerisch, da keine geschlossene Lösung für dieses Problem bekannt ist. Betrachtet man allerdings nur zwei Himmelskörper, so kann man das Modell auch analytisch untersuchen.

Die durch Simulation am Computer untersuchbaren Modelle lassen sich wie folgt charakterisieren:

> **Simulationsmodelle** *im Sinne einer Computersimulation sind abstrakte, numerische, deskriptive und dynamische Modelle. ([Pag94], S. 34)*

Statische Modelle können dabei als einfacher Spezialfall dynamischer Modelle betrachtet werden. Präskriptive Modelle werden in der Regel nicht zur Computersimulation verwendet. Die Schlussfolgerungen und Interpretationen der Simulationsergebnisse müssen nachträglich vom Simulationsmodell auf das System übertragen werden.

Da Simulationsmodelle in einer Vielzahl von verschiedenen Bereichen verwendet werden, bietet es sich an Simulationsmodelle nach weiteren Eigenschaften zu klassifizieren. Für jede dieser zusätzlichen Eigenschaften, die in Tabelle 1 zusammengefaßt sind, läßt sich in eine „leichte“ und „schwere“ Variante im Sinne der Komplexität angeben. Durch die Einteilung in verschiedene Modelltypen wird zum einen die Implementation des Simulationsprogramms erleichtert, da dieser für einen bestimmten Modelltyp geeignete Annahmen machen kann und aufgrund dieser Annahmen einen effizienten Ablauf bereitstellen kann. Zum anderen erleichtert die Bestimmung von weiteren Eigenschaften die Anwendung einer formalen Methode, die in Kapitel 3 beschrieben werden. Viele der Eigenschaften lassen sich nicht nur auf Simulationsmodelle, sondern auch auf Systeme im Allgemeinen im anwenden.

Leichte Variante	Schwere Variante
linear	nichtlinear
stabil	instabil
steady-state	transient
deterministisch	stochastisch
autonom	nicht-autonom
diskret	kontinuierlich

Tabelle 1: Eigenschaften von Simulationsmodellen

Ein ***lineares*** Modell kann durch lineare Funktionen beschrieben werden, ein ***nichtlineares*** Modell beinhaltet komplexere Beziehungen. Ein ***stabiles*** Modell kehrt in seine Ausgangslage zurück, nachdem die Störung des Ausgangszustands behoben ist, was für ***instabiles*** Modell nicht gelten muß. Ein ***transientes*** Modell verändert sein Verhalten mit der Zeit, währen ein ***steady-state*** Modell in jeder Periode das gleiche Verhalten hat. Ein ***stochastisches*** Modell beinhaltet mindestens eine Beziehung, die mit einer zufälligen Variable beschrieben werden muß. ***Autonom*** sind Modelle, die außer der Modellinitialisierung keine weiteren Eingaben von außen erhalten. In einem ***(zustands-)diskreten*** Modell haben alle Zustandsvariablen einen endlichen und diskreten Wertebereich, wobei die Änderung des Wertes zu einem bestimmten Zeitpunkt erfolgt. In einem ***(zustands-)kontinuierlichen*** Modell ändern sich die Werte der Zustandsvariablen kontinuierlich im Verlauf der Zeit, d.h. der Wertebereich ist ein reelles Intervall. Zusätzlich dazu kann noch zwischen ***zeitdiskreten*** und ***zeitkontinuierlichen*** Modellen unterschieden werden, wobei in zeitdiskreten Modellen eine Zustandsänderung nur zu bestimmten, regelmäßigen Zeitpunkten (Takt) stattfinden kann, wohingegen in zeitkontinuierlichen Modellen eine Zustandsänderung zu jedem Zeitpunkt möglich ist. Für zeitdiskrete Modelle ist die Taktdauer eine für Effizienz und Genauigkeit der Simulation entscheidende Größe, die nicht leicht zu bestimmen ist.

Ein schwingenden Pendels kann in einem nichtlinearen, stabilen, steady-state, deterministischen, nicht-autonomen und kontinuierlichen Modell dargestellt werden. Ein lokales Rechnernetzwerk mit der Übertragung von Paketen kann beispielsweise in einem nichtlinearen, instabilen, transienten, stochastischen, autonomen und diskretem Modell abgebildet werden.

Für einige Systeme bietet es sich an, bestimmte Komponenten durch einen Modelltyp und andere Komponenten durch andere Modelltypen zu beschreiben. Daher existieren verschiedene Ansätze, Mischformen zwischen den oben beschriebenen Modelleigenschaften darzustellen. Eine häufig anzutreffende Mischform ist die Verknüpfung von diskreten und kontinuierlichen Modellen. [Ber97] stellt einen solchen Ansatz vor und gibt ein anschauliches Beispiel eines Modells, daß nicht mit einem rein diskreten oder rein kontinuierlichen Modell zufriedenstellend gelöst werden kann. Dabei handelt es sich um den Start einer Rakete von einem Himmelskörper. Erreicht die Rakete nicht die notwendige Fluchtgeschwindigkeit[2], so steigt sie in eine maximale Höhe und fällt zurück zur Oberfläche. Überschreitet sie diese Fluchtgeschwindigkeit, so verläßt die Rakete den Anziehungsbereich und bewegt sich unendlich weit weg vom Startpunkt[3]. Ein diskretes Modell könnte die Ergebnisse „fällt zurück“ und „fliegt unendlich weit weg“ erlangen, nicht aber die maximale Höhe für den Fall des Zurückfallens. Ein kontinuierliches Modell findet zwar die Lösung für diesen Fall, kann aber nicht das Ergebnis „fliegt unendlich weit weg“ finden, da dazu unendlich viele Simulationsschritte benötigt.

Im Bereich der Künstlichen Intelligenz ist oft eine andere Bezeichnungsweise, motiviert durch andere Ziele und Ansätze, anzutreffen. Ereignisse werden in diesem Zusammenhang auch als ***Landmarken (Landmarks)***, Systeme als ***Prozesse***, zustandskontinuierliche Modelle als ***numerische Modelle*** und zustandsdiskrete Modelle als ***qualitative Modelle*** bezeichnet. Statische und dynamische Modelle werden auch ***Ein-Phasen-*** und ***Mehr-Phasen-Simulation*** genannt. [Fis92] und [Pup90] geben eine Einführung in die Simulation aus der Sichtweise der KI.

2.3. Diskrete ereignisgesteuerte Simulation

Die am häufigsten verwendete Architektur eines Simulationsprogramms ist die ***Diskrete Ereignisgesteuerte Simulation (discrete event simulation, DES)***. DES ermöglicht die Simulation von Simulationsmodellen, die nichtlinear, instabil, transient, stochastisch, autonom und zustandsdiskret sind. Damit sind außer den nicht-autonomen und kontinuierlichen Simulationsmodellen alle Modellvarianten durch eine DES simulierbar, so daß DES auf eine sehr große Menge von Simulationsmodellen angewendet werden kann. Die Eigenschaft der Autonomie eines Modells hängt hauptsächlich von der Sichtweise des Modellierers ab, was „innerhalb“ und „außerhalb“ des Modells liegt. Die Simulation kontinuierlicher Simulationsmodelle ist mit der Problematik verbunden, daß in der üblichen Computerarchitektur wirklich kontinuierliche Wertebereiche nicht darstellbar sind, sondern nur approximiert werden können.

Die Systemzustandsvariablen in einer DES sind diskret und ändern sich an verschiedenen Zeitpunkten spontan, d.h. die Änderung selbst hat keine Zeitdauer. Die Zeitpunkte sind im mathematischen Sinne abzählbar und in einem realen Simulationslauf endlich. Eine solche Veränderung des Systemzustands zu einem Zeitpunkt wird als ***Ereignis*** bezeichnet.

Durch Ereignisse werden die Werte der Zustandsvariablen auf den Wertebereich von abschnittsweise definierten und in diesen Abschnitten konstanten Funktionen beschränkt. Betrachtet man einzelne Objekte des Systems, so kann man Ereignisse als an bestimmte Objekte gerichtet auffassen. Für ein Ereignis sind das diejenigen Objekte, deren Zustandsvariablen (die einen Teil des Gesamtsystemzustands darstellen) durch das Ereignis geändert werden. Den Zeitraum zwischen zwei aufeinanderfolgenden, an ein Objekt gerichteten Ereignissen wird als ***Aktivität eines Objektes*** bezeichnet ([Pag94], S.

[2] Fluchtgeschwindigkeit: Mindestgeschwindigkeit, um der Anziehungskraft der Erde zu entkommen (für die Erde: ca. 11 km / s)

[3] Die dritte Möglichkeit des Einschwenkes in eine Umlaufbahn kann in diesem vereinfachten Modell nicht auftreten.

12). Der ***Ereigniskalender (event schedule***) ist die Agenda aller Ereignisse, deren Zeitpunkte bereits bekannt, die aber noch nicht im Laufe der Simulation abgearbeitet worden sind.

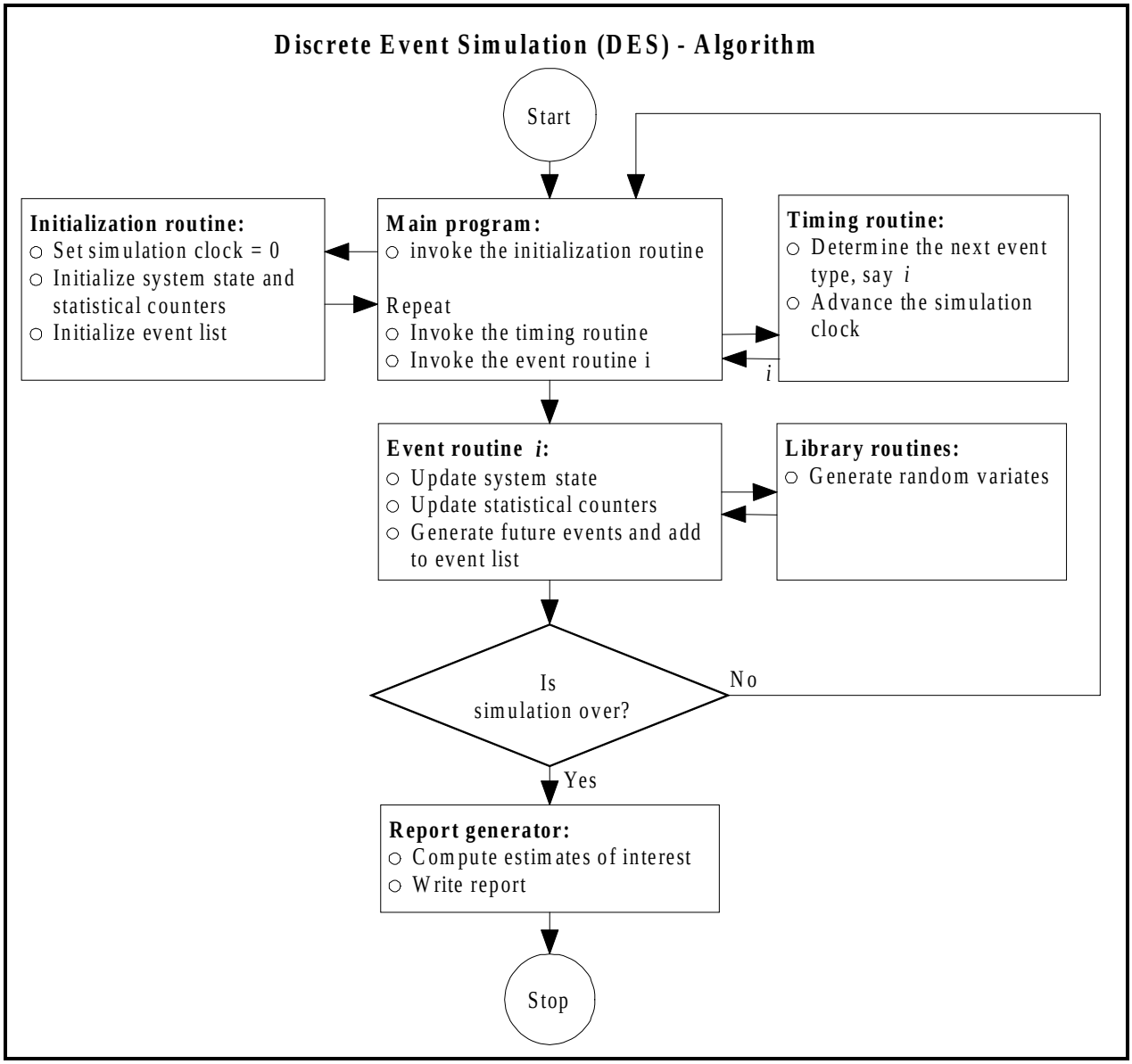


Abbildung 1: Verfahren des kritischen Ereignisses (aus [Law91], S. 12)

Da eine DES ein dynamisches Simulationsmodell simuliert, werden Mechanismen benötigt, um den Ablauf der Zeit darzustellen. Für diesen Ablauf existieren zwei verschiedene Möglichkeiten ([Law91], S 8f):

- ***Verfahren des kritischen Ereignisses (next-event time advance):*** Zu Beginn der Simulation wird die Simulationsuhr initialisiert und die Zeitpunkte der zukünftigen Ereignisse werden berechnet und auf einer Agenda vermerkt. Die Uhr wird dann auf den Zeitpunkt des zeitlich nächsten Ereignisses gesetzt, die Zustandsänderungen dieses Ereignisses werden ausgeführt, neue Ereignisse berechnet und auf die Agenda gesetzt. Das bearbeitete Ereignis wird aus der Agenda gelöscht. Danach wird die Uhr auf den Zeitpunkt des jetzt nächsten Ereignisses gesetzt usw., solange bis ein externes Endkriterium erreicht ist, oder kein Ereignis mehr auf der Agenda vorhanden ist. Die Sprünge der Simulationsuhr sind somit unterschiedlich groß.
- ***Zeitscheibenverfahren (fixed-increment time advance):*** In diesem Ansatz wird die Simulationsuhr in jedem Schritt um einen festen Betrag Δt erhöht. Nach dem Inkrementieren der Uhr wird auf der Agenda nachgeschaut, welche Ereignisse für das letzte Intervall angesetzt waren. Diese Ereignisse werden so behandelt, als würden sie am Ende des Intervalls stattfinden. So-

mit stehen jetzt möglicherweise mehrere Ereignisse zur Ausführung bereit, so daß durch Regeln festgelegt werden muß, in welcher Reihenfolge die Ereignisse abgearbeitet werden. Dieser Ansatz beinhaltet in der Regel kleinere Ungenauigkeiten durch das Nachbearbeiten von Ereignissen, Ineffizienz durch mögliche Leertakte und eine kompliziertere Implementierung durch die Regeln. Außerdem ist Bestimmung der Taktgröße schwierig, da zu große Taktdauern die Ungenauigkeiten vergrößern und zu geringe Taktdauern zu Effizienzverlusten führen. Je nach Art des Systems können diese Nachteile verschwinden und die getaktete Zeitfortschreitung die effizientere Lösung darstellen, wenn sich das System ebenfalls getaktet verhält. Beispielsweise beruhen langfristige Simulationen von Wirtschaftssystemen wie einer Aktienbörse auf getakteten Meßwerten wie der täglichen Kursfestsetzung. Wird eine DES zur Simulation eines kontinuierlichen Modells angewendet, so bietet sich ebenfalls eine getaktete Zeitfortschreitung an. Der getaktete Ansatz ist durch die Einführung von Ereignissen zu den Taktzeitpunkten in das Verfahren des kritischen Ereignisses überführbar.

Das Prinzip einer DES mit Zeitfortschreitung zum nächsten Ereignis ist das in Simulationssprachen und -tools am häufigsten verwendete, da es sowohl einfach zu implementieren als auch effizient ist. Insgesamt besteht ein auf der Architektur einer DES aufbauendes Simulationsprogramm aus dem in Abbildung 1 abgebildeten Algorithmus. Der Algorithmus zur Durchführung einer DES besteht aus drei Phasen: In der Initialisierung wird die Simulationsuhr auf einen Startwert gesetzt und ein Anfangsereignis generiert. Solange noch Ereignisse auf der Ereignisagenda vorhanden sind und das Abbruchkriterium nicht erfüllt ist, wird das nächste Ereignis bestimmt und die Simulationsuhr auf den Zeitpunkt dieses Ereignisses gesetzt. Alle Zustandsänderungen, die sich aus diesem Ereignis ergeben, werden ausgeführt und neue, sich ergebende Ereignisse berechnet. Am Ende der Simulation wird eine statistische Auswertung vorgenommen und eine Ausgabe generiert.

2.4. Ablauf einer Simulationsstudie

Der Erstellungsprozess einer Simulationsstudie ähnelt in vielen Bereichen der Entwicklung von Software, da im Prinzip eine spezielle Art von Software erstellt wird. Daher existieren wie in der Softwareentwicklung mehrere Entwicklungsmodelle für Simulationsstudien, die allerdings in ihrer Grundstruktur sehr ähnlich sind. In diesem Abschnitt soll ein Entwicklungsprozess in 7 Phasen vorgestellt werden, der sich an [Dux95] anlehnt. In [Hoo89] und [Law91] finden sich ähnliche Modelle. In Anlehnung an die Bezeichnungsweise von [Dux95] sind die Phasen der Erstellung einer Simulationsstudie:

1. System analysieren und abstraktes Modell formulieren
2. Daten ermitteln und vorbereiten
3. Simulationsmodell aufbauen
4. Simulationsmodell verifizieren und validieren
5. Experimentieren
6. Ergebnisse interpretieren und analysieren
7. Simulation dokumentieren

Dabei ist zu beachten, daß diese Phasen nicht in einer linearen Reihenfolge bearbeitet werden müssen, sondern daß ähnlich dem „Wasserfallmodell“ des Software-Engineering immer Rücksprünge zur vorigen Phase möglich sind, die dann durchgeführt werden müssen, wenn logische oder implementationsbedingte Fehler aus einer früheren Phase erkannt werden. Im folgenden werden die einzelnen Phasen genauer vorgestellt:

1. **System analysieren und abstraktes Modell formulieren:** Zu Beginn einer Simulationsstudie muß das Ziel der Studie und das zu untersuchende System genau definiert werden. Unter Berücksichtigung des Zieles werden die Komponenten des Systems und die Systemgrenze, d.h. die Bereiche, die nicht mehr zum System gehören, ermittelt. Für jede Komponente wird daraufhin eine Menge von Variablen ermittelt, die zu ihrer Beschreibung wichtig sind. Ebenfalls müssen exogene, d.h. von außen einwirkende, Variablen identifiziert und bewertet werden. Bei

der Variablenauswahl muß eine Beschränkung auf die für das Ziel wichtigen Variablen gefunden werden, da i.a. beliebig viele Eigenschaften zur Beschreibung des Systems und seiner Komponenten gefunden werden können. In einem Fertigungssystem wird z.B. die Farbe einer Maschine keine Rolle spielen, wohl aber deren Kapazität, wohingegen in einem geographischen System die Farbe der Erdoberfläche entscheidend sein kann (da sie die Rückstrahlung der eingestrahlten Sonnenenergie bestimmt). Für das System müssen Leistungsmaße festgelegt werden und diese mit den Variablen verknüpft werden. Beispielsweise ist in einem Fertigungssystem die Auslastung der Maschinen ein Leistungsmaß, das aus dem Anteil der Zeit, in der die Maschine arbeitet, zur Gesamtzeit bestimmt wird. Insgesamt werden durch die Systemanalyse die statischen Eigenschaften des Systems ermittelt, die sich in einem abstrakten Modell beschreiben lassen.

2. **Daten ermitteln und vorbereiten:** Der nächste Schritt ist die Bestimmung des Verhaltens des Systems. Dieses wird dabei durch die Änderungen der Zustandsvariablen beschrieben, so daß herausgefunden werden muß, wie sich diese Variablen im realen System verhalten. Dazu müssen Messungen der Variablen oder anderer Leistungsgrößen, die im vorigen Abschnitt definiert wurden, vorgenommen werden. Bei der Messung muß stets berücksichtigt werden, daß sich das beobachtete System verändern kann, wenn Messungen vorgenommen werden, so daß Meßungenauigkeiten entstehen. Beispielsweise ist es schwierig, die Dauer von manuellen Arbeitsgängen zu messen, da der beobachtete Werker zum einen ein Recht darauf hat, zu erfahren, wenn Daten über ihn erhoben werden, zum anderen er sich aber während der Beobachtung besonders motiviert zeigt. Da i.a. sehr viele Daten erhoben werden müssen, um alle Variablen in einer ausreichenden Genauigkeit abzudecken, ist oft eine Automation der Datenerhebung notwendig. Die ermittelten Daten müssen danach ausgewertet werden. In den meisten Fällen bedeutet dies, für die gemessenen stochastischen Schwankungen des Systems geeignete Verteilungsfunktionen und Parameter zu finden, die den gemessenen Daten am meisten entsprechen. Die Erläuterung der dazu üblichen statistischen Verfahren (z.B. χ^2-Test, Kolmogorov-Smirnov-Test, Maximum-Likelihood-Estimator MLES) würde allerdings den Rahmen dieser Arbeit sprengen, weshalb zur Einführung in dieses Thema auf [Law91] und [Dux95] verwiesen wird. Desweiteren ist es wichtig, Wertebereiche für die Variablen festzulegen, was für die Durchführung von Experimenten benötigt wird. In dieser Phase werden somit den in der vorigen Phase ermittelten statischen Eigenschaften des Systems dynamische hinzugefügt.

3. **Simulationsmodell aufbauen:** Nachdem die statischen und dynamischen Eigenschaften ermittelt worden sind, kann daraus ein Simulationsmodell konstruiert werden. Dazu wird zuerst ein passender Modelltyp bestimmt (s. Abschnitt 2.2), der alle ermittelten Eigenschaften des realen Modells überdeckt, aber keine unnötige Komplexität und Effizienzverluste durch zusätzliche Eigenschaften enthält. Ausgehend von einem Modelltyp sollte daraufhin ein Simulationsmodell in einer formalen Methode erstellt werden (s. Kapitel 3). Die Erstellung eines formalen Modells bedeutet zwar an dieser Stelle einen höheren Aufwand, erlaubt aber die Anwendung analytischer Methoden und erleichtert die Verifikation der Implementierung. Für den Aufbau des Modells eignen sich sowohl ein „top-down“- als auch „bottom-up“-gerichteter Ansatz. Welcher Ansatz gewählt werden sollte, hängt dabei von den Zielen der Studie und von den Eigenschaften des Modells ab.

   Die nächste Entscheidung betrifft die Grundlage der Implementierung. Wird eine existierende Simulationssprache oder Simulationstool gewählt, so muß berücksichtigt werden, wie gut ein Werkzeug mit den ermittelten Modelleigenschaften zusammenpaßt, welche Untersuchungsmöglichkeiten es im Hinblick auf das Ziel der Studie erlaubt und mit welchen zusätzlichen Kosten der Einsatz dieses Werkzeugs verbunden ist. Wird das Simulationsmodell in einem eigenen Simulator in einer allgemeinen Programmiersprache erstellt, kann bei der Implementierung speziell auf das Simulationsmodell eingegangen werden. Einer Eigenimplementierung steht der größere Aufwand gegenüber, der für die Programmierung bzw. Wiederverwendung grundlegender Module wie Zufallszahlenerzeugung oder statistische Auswertung anfällt. Nach diesen Entscheidungen kann die Implementation der Simulation durchgeführt werden.

4. **Simulationsmodell verifizieren und validieren:** Unter Modellverifikation wird das Überprüfen der Implementierung auf logische, semantische und syntaktische Fehler verstanden. Dabei wird die Implementierung mit dem abstrakten Modell verglichen. Die Vorgehensweisen bei der Verifikation eines Simulationsmodells entsprechen denen der Softwareentwicklung, ebenso verhält es sich mit der Komplexität. Der Beweis, daß ein Simulationsmodell korrekt ist, ist ebenso schwierig bis unmöglich zu erbringen wie der Beweis über die Korrektheit von Programmen. Je nach Grundlage der Implementierung wird die Fehlersuche durch Werkzeuge unterstützt. Syntaktische Fehler werden in Simulations- und Programmiersprachen durch Compiler erkannt, ebenso lassen sich in Simulationssprachen manche logische Fehler, wie z.B. nie ausgelöste Ereignisse, erkennen. Die Fehlererkennung wird durch ein formales Modell erleichtert, da für diese analytische Tests zur Verfügung stehen.

   Die Validierung ist die Überprüfung des Modells durch Vergleich mit dem realen Ausgangssystem, bei der gezeigt werden soll, daß das Modell das gleiche Verhalten zeigt. Dazu müssen Daten über das System- und Modellverhalten ermittelt und miteinander verglichen werden. Eine genaue Übereinstimmung zwischen beiden Datenmengen ist allerdings kaum möglich, weshalb man mit einer zu definierenden Ähnlichkeit zufrieden sein muß. Daß es keine genaue Übereinstimmung geben kann, liegt u.a. daran, daß das Modell eine Abstraktion und Idealisierung des Systems ist und daß bei der Messung des Systemverhaltens Meßfehler auftreten. Schwierig ist die Validierung von noch nicht existierenden Systemen, da hierbei nur ein Vergleich mit Annahmen und Schätzwerten, die bereits in das Modell eingegangen sind, möglich ist. Wichtig bei der Validierung ist die Sensitivitätsanalyse, d.h. eine Untersuchung wie stark sich geringe Veränderungen der Anfangsdaten sich auf das Ergebnis auswirken. Falls man davon ausgehen kann, daß das zu untersuchende System nicht „chaotisch“ ist, darf eine geringfügige Veränderung der Eingangsparameter nur eine kleine Veränderung im Systemverhalten zeigen.

   Treten bei der Verifikation oder Validierung des Modells Probleme auf, so sind eine oder mehrere Phasen (bis zur Systemanalyse) zu wiederholen.

5. **Experimentieren:** Durch die bisherigen Phasen ist eine korrekte Implementierung eines Simulationsmodells des Systems entstanden, die als Grundlage für Experimente dient. Alle bisherigen Phasen sind im Prinzip nur Vorarbeiten zu diesem entscheidenden Abschnitt. Zur Durchführung der Experimente ist eine genaue Planung notwendig, da i.a. beliebig viele Experimentationsmöglichkeiten zur Verfügung stehen, so daß eine Auswahl auf das Wesentliche im Hinblick auf die Ziele und die zur Verfügung stehenden Rechenzeit getroffen werden muß.

   Da von einem stochastischen Simulationsmodell ausgegangen wird, ist der Ausgang eines Simulationslaufes ebenfalls mit einer zufälligen Schwankung behaftet. Insbesondere spielt die Varianz der Ausgabe eine entscheidende Rolle, da sie darüber bestimmt, wie aussagekräftig die Ergebnisse sind. Eine einfache, aber ineffiziente Methode, die Varianz der Ausgabe zu senken, ist die Wiederholung von Simulationsläufen mit den gleichen Eingabedaten und die Zusammenfassung der Ergebnisse. In [Law91] wird eine Einführung in diese Methoden, die damit zusammenhängenden statistischen Verfahren und weitere Techniken zur Varianzanalyse (analysis of variance, ANOVA) und Varianzsenkung gegeben. Ebenfalls ist bei der Analyse der Ergebnisse zu beachten, ob die Simulation mit einem „leeren“ System begonnen hat. Ein leeres System besitzt in der Regel eine Einschwingphase, die bei der Analyse entweder entfernt oder mitberücksichtigt werden muß. Die Aufgabenstellungen, eine geeignete Initialisierung des Systems zu finden, so daß keine Einschwingphase auftritt, oder die Größe der Einschwingphase zu ermitteln, sind nicht trivial und oft nur durch Probieren am Modell zu lösen. [Law91] beschreibt statistische Techniken, wie die Länge der Einschwingphase ermittelt werden kann.

   Somit gestaltet sich bereits die Durchführung eines einzelnen Experiments, d.h. die Simulation mit einem Satz von Eingabeparametern, schwierig. Noch komplexer wird allerdings die Untersuchung, welche Parameter auf welche Weise das Systemverhalten beeinflussen. Dazu müssen jeweils Experimente mit unterschiedlichen Eingabeparametern durchgeführt und miteinander verglichen werden. Entscheidend beim Vergleich ist auch hier das Ziel der Simulationsstudie.

So kann es wichtig sein, Parameterbelegungen für ein „optimales“ Systemverhalten zu finden, oder durch verschiedene Parametersätze ein Verständnis für kausale Zusammenhänge innerhalb des Systems zu gewinnen. Einer der Ansätze zur Planung und Durchführung ist das „factorial design“. Dabei werden aus den Eingabeparameter eine zu untersuchende Teilmenge, genannt „Faktoren“, ausgewählt. Für jeden Faktor wird ein zu simulierender Wertebereich definiert und daraufhin alle sich daraus ergebenden Kombinationen in Simulationsläufen getestet. Im einfachsten Fall ergeben sich für jeden Faktor ein Ausgangswert und ein „verbesserter“ Wert, was bei k Faktoren bereits zu 2^k Kombinationen führt. Dies zeigt, daß man sich auf eine geringe Teilmenge der Eingangsparameter beschränken muß. Für die Auswertung von solchen Experimenten und dem Vergleich von unterschiedlichen Simulationsläufen, u.a. auf die Fragestellung hin, ob eine „Verbesserung“ die gewünschten Ergebnisse nachweislich zeigt, existieren verschiedene statistische Verfahren. In [Law91] wird eine Übersicht zu diesen Verfahren gegeben.

6. **Ergebnisse interpretieren und analysieren:** Die durch die Experimente gewonnenen Ergebnisse müssen schließlich noch interpretiert werden. Die Interpretation muß die Ergebnisse vom Simulationsmodell auf das zugrundeliegende System transferieren und dabei berücksichtigen, welche Annahmen bei der Modellierung getroffen wurden. Als Analysemöglichkeit bieten sich unter anderem graphische Verfahren an, um die Abhängigkeiten mehrerer Parameter zu veranschaulichen. Nach einer genauen Interpretation der Ergebnisse können diese dann im realen System umgesetzt werden. Da nicht immer die „beste“ Lösung implementiert werden kann, müssen stets mehrere Möglichkeiten berücksichtigt werden, die dann zur Entscheidungsfindung präsentiert werden.
7. **Simulation dokumentieren:** Das Dokumentieren der Simulationsstudie ist eine Phase, die nicht nur am Ende erfolgen sollte, sondern sich durch den kompletten Verlauf zieht. Zum einen dient eine genaue Dokumentation dem Wiederholen der Simulationsstudie zu einem späteren Zeitpunkt, wobei frühere Simulationstudien des gleichen Systems zum Vergleich dienen. Einzelne Bausteine der Implementierung können in anderen Simulationsstudien wiederverwendet werden. Das bedeutet, daß die Implementation immer so allgemein wie möglich gehalten werden soll, damit eine Wiederverwendung von Komponenten möglich ist. Ebenso sollten alle Probleme, die im Verlauf der Simulationsstudie auftraten, beschrieben werden, um diese beim nächsten Mal zu vermeiden.

Eine Simulationsstudie beginnt somit mit der genauen Definition des zu untersuchenden Systems und der Ziele, die in dieser Studie erreicht werden können. Erst nach einer Analyse der Bestandteile und des Verhalten des Systems kann ein abstraktes Modell formuliert werden. Das abstrakte Modell ist daraufhin zu untersuchen, wie es am besten zu implementieren ist. Die Implementation muß gründlich verifiziert und validiert werden, bevor mit der Durchführung von Experimenten am System begonnen werden kann. Dabei ist eine Planung der Experimente sowie eine genaue Analyse der Ergebnisse auf ihren Inhalt und ihre Aussagesicherheit hin notwendig. Die aus den Experimenten geschlossenen Folgerungen können abschließend am System umgesetzt werden.

2.5. Zusammenfassung

Ausgehend von einem zu untersuchenden System und den Untersuchungszielen wird ein Modell des Systems erstellt und simuliert. Dabei sind die durch Computer simulierbaren Modelle dadurch charakterisiert, daß sie abstrakte, numerische, deskriptive und dynamische Modelle sind. Mit dem Konzept der DES steht ein einfacher und erweiterbarer Grundalgorithmus zur Simulation dieser Modelle zur Verfügung. Ein an die Methoden des Softwareentwurfs angelehntes Ablaufkonzept erleichtert die Planung, Durchführung und Überprüfung einer Simulationsstudie.

Für die Simulation einer Fertigung kann davon ausgegangen werden, daß ein geeignetes Modell nichtlinear, stabil, stochastisch, zeitkontinuierlich und nicht-autonom ist. Die Stabilität ergibt sich aus der Tatsache, daß der Start einer Bearbeitung eine Maschine aus ihrem Ruhezustand bringt und nach der Beendigung eines Bearbeitungsvorgangs dieser Zustand wieder angenommen wird. Da eine Ferti-

gung über verschiedene Wege mit der Außenwelt in Kontakt steht, ist das Modell nicht-autonom. Bei der Modellerstellung müssen beispielsweise der Materialan- und -abstransport berücksichtigt werden. Erfolgt die Fertigung auftragsorientiert, so liegt ein transientes Modell vor. Bei einer Massenfertigung, die auf Lager produziert, kann auch ein steady-state Modell angewendet werden. Wird die Fertigung sehr detailliert modelliert, so kann die Berücksichtigung kontinuierlicher Eigenschaften notwendig werden. Für einen höheren Abstraktionsgrad, wie er sich bei der Modellierung des Informationsflusses durch die Fertigung ergibt, genügt ein diskretes Modell, das mit DES-Algorithmus simuliert werden kann. Der Informationsfluß kann dabei durch einzelne Ereignisse, die den Start und das Ende einzelner Arbeitsschritte darstellen, und diskrete Zustände, die anzeigen, ob eine Maschine arbeitet oder ruht, erfolgen.

3. Formale Methoden zur Modellierung

Um Modelle erstellen, verifizieren und simulieren zu können, benötigt der Modellierer Methoden, die es ihm erlauben, sich auf das Wesentliche des Modells zu konzentrieren und dieses auch anderen zur Beurteilung vermitteln zu können. Die Methoden zur Modellerstellung lassen sich in zwei große Klassen einteilen:

- Simulationssprachen und -shells
- Formale Methoden

Simulationssprachen und Simulationstools legen ihren Schwerpunkt auf die Simulation des Modells und die statistischen Auswertungen der Simulationsläufe. Formale Methoden stellen die analytische Verifikation des Modells in den Vordergrund. Beide Methoden haben ihre Vor- und Nachteile, die in Tabelle 2 zusammengefasst sind.

Simulationssprachen und -shells	Formale Methoden
implementationsabhängig	implementationsunabhängig
konkret	abstrakt
statistisch auswertbar	analytisch untersuchbar
effizienter	i.d.R. ineffizienter

Tabelle 2: Gegenüberstellung von Simulationssprachen und Formalen Methoden

Unter einer formalen Methode ist eine Methode zu verstehen, die unabhängig von einer Programmiersprache oder Implementierung dargestellt werden kann. Im Prinzip läßt sich mit einer formalen Methode ein Simulationsmodell „auf dem Papier“ erstellen, ohne Computer als Hilfsmittel zu benutzen. Trotzdem stellt die Entwicklung und Bereitstellung rechnerbasierter Werkzeuge bei der Modellierung, Analyse und Simulationsdurchführung ein wichtiges Merkmal fast aller der vorgestellten formalen Methoden dar.

Allen Methoden ist gemeinsam, daß sie in irgendeiner Weise den Begriff eines ***Systems*** einführen, das sich in einem bestimmten ***Zustand*** befindet. Was das System ausmacht und wie sich ein Zustand ändern kann, ist je nach Sichtweise und Intention der Methode verschieden.

In den Abschnitten 3.1 bis 3.3 werden drei Modellierungsmethoden - Ereignisgraphen, DEVS und Petri-Netze - erläutert. In Abschnitt 3.4 und 3.5 werden weitere Modellierungsmethoden vorgestellt, die aber nur kurz umrissen werden. Abschnitt 3.6 bietet eine Zusammenfassung und eine kurze Bewertung der vorgestellten Methoden, v.a. im Hinblick auf das Problem der Simulation einer Fertigung. In Kapitel 5 wird auf die Modellierung mit Hilfe von Simulationswerkzeugen und -sprachen eingegangen.

3.1. Event Graphs (Ereignisgraphen)

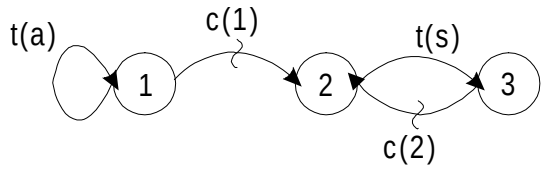
Ereignisgraphen stellen eine graphische Methode dar, um Modelle für DES darzustellen. Sie wurden zum ersten Mal 1983 von L. Schruben vorgestellt und seitdem mehrmals erweitert. Die folgende Einführung in Ereignisgraphen ist aus [Bus96] und [Pag94] entnommen.

3.1.1. Definition

Die grundlegenden Elemente einer DES sind Zustandsvariablen und Ereignisse. Zustandsvariablen beschreiben den Zustand des Systems, während durch Ereignisse die Zustandsvariablen geändert werden. Die logischen und zeitlichen Beziehungen zwischen Ereignissen können durch einen Ereignisgraphen dargestellt werden.

Ein Ereignisgraph ist ein gerichteter Graph. Jeder Knoten stellt ein Ereignis dar. Es existieren zwei Klassen von Kanten. Eine durchgezogene Kante zwischen Knoten *i* und *j* bedeutet, daß das Ereignis *i* zur Folge haben kann, daß Ereignis *j* ausgelöst wird (***scheduling***). Wenn die Auslösung von Ereignis *j*

an eine Bedingung geknüpft ist, so wird die Kante unterbrochen dargestellt und die Bedingung an der Kante vermerkt. Die Bedingung wird zum Ereigniszeitpunkt des Ereignisses *i* ausgewertet. Falls zwischen den beiden Ereignissen ein zeitliches Intervall liegt, so wird dies am Ausgang der Kante durch die Angabe des Intervalls oder einer zufälligen Variable beschrieben.

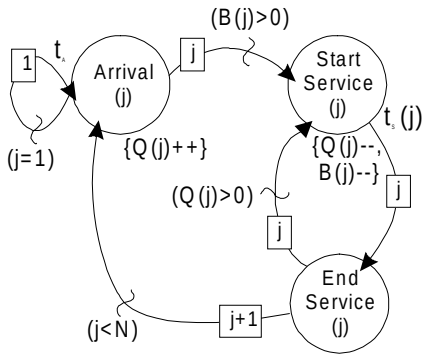


Zustandsvariable	
n	Anzahl der Anforderungen im System
Ereignisse	
1	Eintreffen einer Anforderung
2	Start der Bedienung
3	Ende der Bedienung
Bedingungen	
c(1)	n=0
c(2)	n>0
Verzögerungen	
t(a)	Zwischenankunftszeit
t(s)	Bedienzeit

Abbildung 2: Warteschlange mit Bedieneinheit (aus [Law91], S. 46)

Die zweite Klasse von Kanten wird durch eine gestrichelte Linie dargestellt und bedeutet die Stornierung eines Ereignisses (***cancelation***). Auch hier kann eine Bedingung angegeben werden. Zu jedem Ereignis kann eine Übergangsfunktion vermerkt werden, die die Änderung der Zustandsvariablen durch das Ereignis steuert. Fehlt diese Funktion, so ändert sich der Systemzustand nicht. Wenn nur der Zusammenhang zwischen Ereignissen abgebildet werden soll, wird die Übergangsfunktion nicht dargestellt. Sind Bedingungen und Übergangsfunktion kurz genug, so werden sie direkt in den Graphen geschrieben, ansonsten wird nur ein Index dargestellt. Konventionsgemäß werden Bedingungen in runden, Auswirkungen in geschweiften Klammern geschrieben.

Es ist erlaubt, daß ein Ereignis mehrere (auch kein) Nachfolgeereignisse und daß ein Ereignis sich selbst als Nachfolger hat. Auch mehr als eine Kante von einem Ereignis zum nächsten sind möglich. Die Simulation verläuft nach dem Grundprinzip der DES, indem das nächste Ereignis von einer Ereignisliste (Agenda, event list) genommen wird, die Zustandsvariablen gemäß der Übergangsfunktion verändert werden und alle Nachfolgeereignisse erzeugt und auf die Ereignisliste gesetzt werden. Dies wird solange fortgesetzt, bis die Ereignisliste leer ist. Dieses Vorgehen hat zur Folge, daß die Ereignisliste mit einem Ereignis vorbelegt werden muß, damit die Simulation gestartet werden kann. Je nach Konvention wird dies durch einen Pfeil auf das Ereignis oder durch das spezielle Ereignis „Run“ angedeutet. Abbildung 2 zeigt ein Beispiel eines Ereignisgraphen für ein Warteschlangenmodell mit einer Bedieneinheit und einer Warteschlange.



Zustandsvariable	
Q(j), j=1..N	Anzahl der wartenden Anforderungen in Gruppe *j*
B(j), j=1..N	Anzahl der freien Bedieneinheiten in Gruppe *j*
Ereignisse	
Arrival(j)	Eintreffen einer Anforderung bei Gruppe *j*
Start Service(j)	Start der Bedienung bei Gruppe *j*
End Service(j)	Ende der Bedienung bei Gruppe *j*
Verzögerungen	
t_A	Zwischenankunftszeit
$t_S(j)$	Bedienzeit in Gruppe *j*

Abbildung 3: Event Graph for Transfer Line Model (aus [Bus96], S. 158)

Obwohl Ereignisgraphen einen minimalistischen Ansatz mit einer Klasse von Knoten, zwei Klassen von Kanten und zwei Optionen für die Kanten darstellen, kann gezeigt werden, daß mit ihnen jedes Simulationsmodell für eine DES erstellt werden ([Bus96], S.154).

3.1.2. Erweiterungen

Um die Modellerstellung und die Lesbarkeit zu erleichtern, existieren die folgenden Erweiterungen zu Ereignisgraphen:

- **Verwendung von Datenstrukturen:** Anstelle von Zustandsvariablen werden Datenstrukturen, z.B. Klassen im Sinne einer objektorientierten Modellierung verwendet.
- **Markierung von Knoten:** ähnlich wie in Petri-Netzen (s. Abschnitt 3.3) können die Knoten mit Marken versehen werden. Diese geben an, wieviele Ereignisse von diesem Typ noch nicht abgearbeitet worden sind.
- **Attributübergabe:** eine Kante kann zusätzlich zu Verzögerung und Bedingung eine Menge von Attribute enthalten. Diese werden beim Auslösen des Ereignisses berechnet und an das nächste Ereignis übergeben. Dargestellt wird dies durch ein Kästchen in der Kante, das die Attribute enthält. [Bus96] stellt ein Beispiel mit einer Menge von *N* Bediengruppen mit jeweils *B[j], j=1..N* identischen, seriell angeordneten Bedieneinheiten dar. Eine Anforderung wird zuerst von einer Bedieneinheit in Gruppe 1, danach von einer Bedieneinheit in Gruppe 2 usw. bedient. Durch die Attributübergabe wird eine *N*-malige Wiederholung des gleichen Sachverhalts vermieden. Als Attribut *j* dient dabei die Nummer der Bediengruppe, an die das Ereignis gerichtet ist. Abbildung 3 zeigt das vorgestellte Beispiel.
- **Simulationsgraphen:** [Pag94] stellt eine Formalisierung von Ereignisgraphen vor, die als Simulationsgraph bezeichnet wird. Dabei läßt sich ein Simulationsgraph formal durch das Quadrupel $G = \left(V(G), E_S(G), E_C(G), Y_G\right)$ beschreiben, wobei

  – $V(G)$ *die Menge der Ereignisknoten,*

  – $E_S(G)$ *die Menge der auslösenden Ereigniskanten (scheduling edges),*

  – $E_C(G)$ *die Menge der stornierenden Ereigniskanten (canceling edges) und*

  – Y_G *die Inzidenzrelation*

  darstellt. Zusammen mit den folgenden indizierten Mengen

  – $F = \left\{ f_v : STATES \to STATES \middle| v \in V(G) \right\}$, *der Menge der Zustandsübergangsfunktionen*

  – $C = \left\{ C_e : STATES \to \{0,1\} \middle| e \in E_S(G) \cup E_C(G) \right\}$, *der Menge der Kantenbedingungen*

  – $T = \left\{ t_e : STATES \to [0,\infty[\middle| e \in E_S(G) \right\}$, *der Menge der Verzögerungen*

  – $\Gamma = \left\{ \gamma_e : STATES \to [0,\infty[\middle| e \in E_S(G) \right\}$, *der Menge der Ereignisprioritäten*

  bildet das Tupel $S = (F, C, T, \Gamma, G)$ das Simulationsgraphmodell (simulation graph model, SGM)

Aus einem SGM läßt sich durch Hinzufügen von Knoten ein expandiertes Simulationsgraphmodell (expanded simulation graph model, ESGM) ableiten, indem in jedem Knoten maximal eine Zustandsvariable verändert wird und jede Bedingung nur aus einer Relation, nicht aber aus einem aussagenlogischen Ausdruck besteht. Das Verhalten bleibt dabei erhalten. Ein so abgeleitetes ESGM ist nicht eindeutig, aber die möglichen Lösungen sind jeweils isomorph (im graphentheoretischen Sinn) zueinander und bilden eine Äquivalenz-Klasse. Das bedeutet, daß man zwei Simulationsmodelle anhand ihrer ESGM auf Gleichheit überprüfen kann.

3.1.3. Analytische Auswertung

Die im vorigen Abschnitt vorgestellten Simulationsgraphen lassen sich als ein Komplexitätsmaß für Simulationsmodelle benutzen. Dabei werden in [Pag94] die folgenden drei Maße vorgestellt:

- $C_1 = |V(G)|$ *(Anzahl der Knoten),*
- $C_2 = \frac{|E(G)|}{|V(G)|}$ *(relative Anzahl der Kanten)*
- $C_3 = \sum_{v \in V(G)} C(v) + \sum_{v,w \in V(G)} I(v,w)$ *(Komplexität der Knoten und Kanten),*

  wobei C(v) die Komplexität des Knotens v und I(v,w) die Komplexität der Kante zwischen v und w darstellt.

Für C_3 gibt es allerdings keine klaren Vorgaben für *C* und *I*. Hier kann u.a. die Laufzeitkomplexität von Vorbedingung und Zustandsänderung verwendet werden. Durch eine geeignete Wahl kann eine Laufzeitbestimmung für einen Simulationslauf vorgenommen werden. Ob Eigenschaften wie die Erreichbarkeit von Zuständen in Simulationsgraphen überhaupt entscheidbar sind, ist ein ungelöstes Problem ([Pag94], S. 60).

3.1.4. Zusammenfassung

Ereignisgraphen stellen eine sehr einfache, effiziente Methode zur Modellierung von diskreten ereignisgesteuerten Simulationsmodellen dar. Sie bestehen aus einer sehr kleinen Menge von Elementen, wodurch diese Methode schnell erlernbar ist. Aus diesem Grunde werden Ereignisgraphen häufig in Lehrbüchern über Simulation vorgestellt (z.B. [Hoo89], [Law91]).

Die Modelle werden in einer graphischen Form präsentiert, die den zeitlichen und logischen Zusammenhang zwischen den auftretenden Ereignissen wiedergibt. Die Ereignismodellierung steht bei dieser Methode im Vordergrund, die Modellierung der Zustände wird im Gegensatz zu Petri-Netzen (s. Abschnitt 3.3) nur implizit unterstützt. Dabei stützen sich Ereignisgraphen voll auf das Prinzip der DES. Eine hierarchisches Modellieren ist möglich, wird aber in den meisten Publikationen ([Pag94], [Bus96]) nicht weiter erläutert. Es existieren Werkzeuge für die Modellierung und Simulation mit Hilfe von Ereignisgraphen in Verbindung mit programmiersprachlichen Konstrukten für die Zustandsvariablen, z.B. SIGMA ([Bus96], S.160).

Für die Simulation eines Fertigungssystems eignen sich Ereignisgraphen insoweit, als daß sich das Verhalten der einzelnen Komponenten (z.B. Maschinen) sehr gut durch die Abfolge von Ereignissen beschreiben läßt. So hat z.B. das Ereignis „Maschine beginnt Operation“ das Ereignis „Maschine beendet Operation“ zur Folge, was wiederum den Start der nächsten Operation auslöst. Aufgrund der vielen Einzelkomponenten (Maschinen, Maschinengruppen usw.) ist allerdings eine hierarchischer Aufbau des Modells notwendig, da ansonsten das Modell sehr schnell unübersichtlich wird. Ebenfalls ungünstig ist die Darstellung des Zustands einer Komponente durch seine Attribute. Da in einem Fertigungssystem für jede Komponente sehr viele Attribute vorhanden sind (s. Abschnitt 4.2.1), wird eine Komprimierung des Zustands auf für die Simulation relevanten Attribute benötigt.

3.2. Discrete Event System Specification (DEVS)

Die Methode der Discrete Event System Specification (DEVS) wurde von Bernard Zeigler 1976 entwickelt und ist an die Allgemeine Systemtheorie angelehnt. In dieser mathematischen Theorie werden die Struktur und das Verhalten von dynamischen Systemen behandelt. Die Struktur wird dabei durch einen hierarchischen Aufbau von Teilsystemen bestimmt, die sich untereinander in einer bestimmten Weise beeinflussen. Das Verhalten eines Systems kann durch kausale oder empirische Repräsentationen be-

stimmt werden. Die Repräsentationsformen können für verschiedene Komponenten eines Systems gemischt verwendet werden, da eine Komponente nur durch Ein- und Ausgabe mit anderen Komponenten in Verbindung steht, nicht aber durch interne Vorgänge. Kausale und empirische Repräsentation entsprechen einer „White-Box“ bzw. „Black-Box“-Sicht auf die Komponente. Die folgende Einführung in DEVS ist im wesentlichen aus [Zei93] und [Pag94] entnommen.

3.2.1. Definition

In der allgemeinen Systemtheorie wird ein deterministisches System $\langle T, U, Y, S, \Omega, \delta, \lambda \rangle$ wie folgt definiert ([Fis92], S. 57):

- *T ist die Zeitmenge, z.B. die Menge der reellen oder natürlichen Zahlen*
- *U ist die Eingabemenge, die die möglichen Systemeingaben enthält*
- *Y ist die Ausgabemenge*
- *S ist die Menge der Zustände*
- $\Omega \subseteq \{ f \mid f : T \to U \}$ *ist die Menge der zulässigen Eingabefunktionen*
- $\delta : S \times T \times T \times \Omega \to S$ *ist die Übergangsfunktion*
- $\lambda : T \times S \to Y$ *ist die Ausgabefunktion*

Ein System ist ***zeit-invariant***, falls die Übergangsfunktion nicht von dem absoluten Wert der Zeit abhängt. In diesem Fall reduziert sich die Übergangsfunktion auf $\delta : S \times T \times \Omega \to S$. Dann bedeutet $\delta(s,t,\omega)=s'$, daß sich das System im Zustand *s* befindet und für die Dauer *t* die Eingabe ω anliegt, woraufhin das System in Zustand *s'* übergeht. Eine weitere Vereinfachung der Übergangsfunktion ist die „eingabefreie“ Übergangsfunktion, die nur von Zustand und Dauer abhängt.

Ausgehend von diesem Systembegriff werden in DEVS Modelle definiert, die hierarchisch zu größeren Modellen ausgebaut werden können. Dazu wird zum einen das Konzept der diskreten Ereignisse durch die Angabe einer Zeitfunktion und zum anderen die Idee eines modularen Aufbaus durch die Definitionen von internen und externen Übergangsfunktionen eingebracht. Die einfachen Modelle in DEVS sind ***atomare Modelle (atomic models)***, die keine weiteren Komponenten enthalten. Formal ist ein atomares Model $M = \langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, t_a \rangle$ wie folgt definiert ([Pag94], S. 46):

- *X ist die Menge der Eingabeereignisse.*
- *S ist die Menge der Zustände.*
- *Y ist die Menge der Ausgabeereignisse.*
- $\delta_{int} : S \to S$ *ist die interne Übergangsfunktion.*
- $\delta_{ext} : Q \times X \to S$ *ist die externe Übergangsfunktion.*
  Dabei ist $Q = \{ (s,e) \mid s \in S, 0 \le e < t_a(s) \}$ *der Gesamtzustand von M*
- $\lambda : S \to Y$ *ist die Ausgabefunktion.*
- $t_a : S \to R_0^+$ *die Zeitfortschrittsfunktion (time advance function).*

Der Zustand eines Modells wird durch Zustandsvariablen beschrieben. Das Verhalten des Modells wird durch zwei Klassen von Ereignissen gesteuert. Eingabeereignisse führen zu einem Zustandsübergang, der durch die externe Übergangsfunktion bestimmt ist. Interne zeitlich bestimmte Ereignisse (time scheduled internal events) werden durch die Zeitfortschrittsfunktion ausgelöst. Wenn ein Modell die durch diese Funktion definierte Zeitdauer in einem Zustand verbracht hat, wird ein internes Ereignis ausgelöst und der Zustand verändert sich gemäß der internen Übergangsfunktion.

Um aus atomaren Modellen komplexere Systeme zu konstruieren, wird in DEVS ein ***gekoppeltes Modell (coupled model)*** eingeführt. Diese fassen atomare Modelle zusammen. Ein gekoppeltes Modell $DN = \left\langle D, \{M_i\}, \{I_i\}, \{Z_{i,j}\}, SELECT \right\rangle$ ist definiert durch ([Pag94], S.47):

- *die Menge der Komponentennamen D*
- *für alle i in D ist M_i ein atomares DEVS-Modell für die Komponente i und I_i die Menge der Komponenten, die Eingaben an die Komponente i machen können.*
- *für alle j in I_i ist $Z_{i,j}: Y_i \rightarrow X_j$ die Ausgabeübersetzungsfunktion von Komponente i nach j*
- *SELECT ein Entscheidungsselektor mit $E \subseteq D \rightarrow D, \forall E \neq \emptyset: SELECT(E) \in E$*

Durch diese Definition werden die Ein- und Ausgaben der Einzelmodelle miteinander verknüpft. Das auf diese Weise entstandene Modell hat wiederum eine Ein- und Ausgabe, so daß es selbst in einem gekoppelten Modell verwendet werden kann, ohne daß dort eine Unterscheidung möglich oder notwendig ist, ob das Modell ein atomares oder gekoppeltes ist. Durch die Trennung zwischen innerem Verhalten und Ausgabeverhalten eines Modell ist es möglich, atomare Modelle durch ähnliche Konzepte wie Endliche Automaten (FSM) oder Zelluläre Automaten (s. Abschnitt 3.4) zu implementieren und aus diesen gekoppelte Modelle zu erstellen. [Zei93] demonstriert an einem Beispiel, wie eine Simulation des Wasserabflusses in einer Landschaft in DEVS implementiert wurde. Dazu wird die Landschaft durch einen 3-dimensionalen Würfel von zellulären Automaten dargestellt, wobei jeder Automat eine Region mit Gestein, Wasser, Boden, Oberfläche, Luft, usw. mit verschiedenem Niederschlags-, Verdunstungs- und Transportverhalten darstellt. Jeweils benachbarte Zellen tauschen dann Informationen über den Wassertransport aus, so daß mit der Zeit ein Abbild des Wasserkreislaufs in dieser Landschaft entsteht.

Im Laufe der Zeit wurden Erweiterungen entwickelt, die zusätzlich zu einer zustandsdiskreten Darstellung kontinuierliche Zustände (DESS) und Mischformen (DEV & DESS) erlauben. [Zei93] stellt diese Erweiterungen vor und gibt eine Einordnung innerhalb des Systembegriffs. In DEVS steht nicht direkt die Simulation, sondern die Beschreibung des Systems als „Simulationswissen“ im Vordergrund. DEVS eignet sich aufgrund seines modularen und hierarchischen Aufbaus zu einem objektorientierten Modellieren und ist mehrmals implementiert worden, u.a. in Scheme, CLOS und C++.

3.2.2. Zusammenfassung

DEVS stellt einen Modellierungsansatz dar, der über die reine Erstellung von Simulationsmodellen hinausgeht. In DEVS wird der Ansatz der Wissensmodellierung in den Vordergrund gestellt, um Modelle zu erhalten, die auf eine Vielzahl von quantitativen und qualitativen Fragestellungen eine Antwort geben sollen. Das Haupteinsatzgebiet für DEVS-Modelle ist dabei die Simulation zur Vorhersage des Systemverhaltens. Ebenso können an DEVS-Modelle Diagnose- oder Erklärungsfragen gestellt werden ([Fis92], S74). Durch die Betonung des hierarchischen, modularen Modellierens lassen sich Systeme auf verschiedenen Abstraktionsstufen betrachten. Durch die Anlehnung an die Systemtheorie steht diese Methode auf einer festen theoretischen Grundlage, die in vielen Bereichen angewendet werden kann. DEVS ist nicht auf einen Bereich von Modellen spezialisiert, hat aber den Nachteil, daß es über keine explizite graphische Repräsentation und nur über wenig theoretische Analysemöglichkeiten verfügt. Als reines mengentheoretisches Konzept kann es nicht verwendet werden, für den Einsatz werden weitreichende Computerwerkzeuge zur Modellierung und Simulation benötigt. Diese Werkzeuge sind allerdings bereits in mehreren Sprachen implementiert worden.

Für die Anwendung von DEVS auf die Simulation eines Fertigungssystems ergibt sich hier im Vergleich zu Ereignisgraphen (s. Abschnitt 3.1.4) eine Umkehrung der Vor- und Nachteile. DEVS bietet sehr gute Möglichkeiten, das Fertigungssystem übersichtlich in seine einzelnen Komponenten (z.B. Maschinen, Maschinengruppen, Werkstätten, Standorte usw.) zu gliedern. Es wird aber zusätzlich noch eine Methode benötigt, welche die atomaren Modelle (in diesem Fall einzelne Maschinen) darstellt.

3.3. Petri-Netze

3.3.1. Definition

Petri-Netze stellen einen graphentheoretischen Ansatz dar, den Informationsfluß eines Systems zu modellieren. Sie wurden zum ersten Mal 1962 von Carl Petri vorgestellt und im Laufe der Zeit mit verschiedenen Erweiterungen versehen. Im folgenden wird eine informelle Definition vorgestellt, die aus [Pag94] entnommen ist. Eine formale Definition findet sich u.a. in [Ros83].

Ein Petri-Netz ist definiert als ein bipartiter, gerichteter Graph. Die beiden Teilmengen der Knoten werden ***Zustände (Bedingungen, Stellen, Plätze, places)*** und ***Ereignisse (Transitionen, Aktionen, transitions)*** genannt. Konventionsgemäß werden Zustände durch Rechtecke und Ereignisse durch Kreise dargestellt. Kanten können nur von einem Zustand zu einem Ereignis bzw. umgekehrt, aber niemals von Zustand zu Zustand oder von Ereignis zu Ereignis führen. Kein Knoten darf isoliert sein, d.h. er besitzt keine Kante, und zwischen je zwei Knoten darf maximal eine Kante sein. Führt eine Kante von einem Zustand *Z* zu einem Ereignis *E*, wird Z als ***Eingangszustand*** für *E* und *E* als ***Ausgangsereignis*** für *Z* bezeichnet. Analog dazu sind ***Ausgangszustand*** und ***Eingangsereignis*** definiert. Genauso spricht man von ***Eingangskanten*** und ***Ausgangskanten***.

Jeder Zustand ist zu einem bestimmten Zeitpunkt realisiert oder nicht realisiert. Dies wird durch die Belegung des Zustands mit einer ***Marke*** (***Deut, token)*** angezeigt. Traditionell werden Marken durch einen kleinen schwarzen Punkt angezeigt. Die Verteilung aller Marken auf Zustände wird als ***Markierung*** des Petri-Netzes bezeichnet. In der einfachsten Fassung kann in einem Zustand nur maximal eine Marke sein, dies nennt man auch ***Ein-Marken-Petri-Netz***. Viele Autoren (z.B. [Jen92-1], [Pag94]) lassen diese Einschränkung aber schon zu Beginn fallen und erlauben beliebig viele Markierungen in einem Zustand (s. Abschnitt 3.3.3). Abbildung 4 zeigt ein Beispiel einer Markierung eines Petri-Netzes mit drei Zuständen (*A,B*) und zwei Ereignissen (*1,2*), wobei A und B realisiert sind.

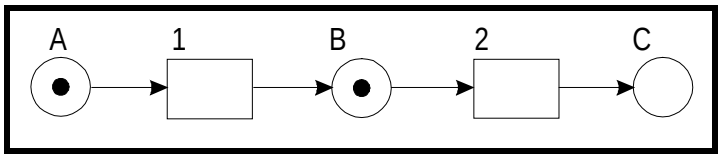


Abbildung 4: Einfaches Petri-Netz

Ein Ereignis heißt ***aktiviert*** ([Ros83], S.18), falls

- *alle Eingangszustände markiert sind und*
- *alle Ausgangszustände markenfrei sind*

Wenn ein Ereignis aktiviert ist, dann *kann* es stattfinden ***(feuern)*** und die ***Schaltregel (Transitionsregel, Simulationsregel, firing-rule)*** wird angewendet. Diese besagt, daß alle Marken aus den Eingangszuständen entfernt und alle Ausgangszustände mit einer Marke belegt werden. Der Übergang ist in einfachen Petri-Netzen als zeitlos anzusehen. Ist die Anzahl der Eingangs- und Ausgangszustände ungleich, so können nach dem Feuern mehr oder weniger Marken im Netz vorhanden sein als vor dem Feuern des Ereignisses. Können in einem Petri-Netz mehrere Ereignisse feuern, so ist die Reihenfolge nicht festgelegt, da nicht jedes aktivierte Ereignis unbedingt feuern muß. Ein Petri-Netz ist somit nichtdeterministisch. Weil das Feuern eines Ereignisses nur von der Markierung der Eingangs- und Ausgangszuständen abhängt, zeigt ein Petri-Netz ein streng lokales Verhalten. In zwei Sonderfällen muß die Schaltregel erweitert werden:

- Ein Ereignis ohne Eingangszustände kann schalten, wenn seine Ausgänge nicht markiert sind.
- Ein Ereignis ohne Ausgangszustände kann feuern, wenn seine Eingänge markiert sind.

Eine weitere Eigenschaft von Petri-Netzen ist, daß sie hierarchisch aufgebaut werden können. Man kann einen zusammenhängenden Teilgraphen durch einen detaillierteren oder vereinfachten ersetzen, ohne die Funktionsweise des restlichen Netzes zu verändern. Dies liegt in der rein lokalen Schaltungsregel begründet. Das einzige, was bei der Verfeinerung oder Vergröberung des Petri-Netzes beachtet werden muß, ist, daß alle Ein- und Ausgangskanten erhalten bleiben. [Jen92-1] gibt Beispiele wie man programmiersprachliche Konstrukte wie Blöcke, Verzweigungen und Schleifen als Petri-Netze darstellen kann und somit durch Schachtelung dieser Elemente „programmieren“ kann ([Jen92-1], S.36-39).

Die Simulation eines Petri-Netzes verläuft relativ einfach. Zu Beginn wird das Netz mit einer Anfangsmarkierung belegt. In jedem Schritt werden alle schaltbaren Ereignisse ermittelt, eines von ihnen wird ausgewählt und ausgeführt. Dieser Schritt wird solange wiederholt, bis entweder kein Ereignis mehr feuern kann oder ein Stopkriterium erfüllt ist (z.B. eine feste Anzahl von Schritten).

Zwei Punkte bei der Simulation verdienen eine nähere Erläuterung: Wenn in einem Schritt mehrere Ereignisse feuern können, so muß für eine Simulation eine Auswahl getroffen werden, da das nichtdeterministische Verhalten von Petri-Netzen nicht simuliert werden soll. Die Auswahl kann rein zufällig oder durch die Angabe von exogenen Regeln erfolgen. Wenn die Simulation dadurch endet, daß kein Ereignis mehr feuern kann, so kann dies mehrere Ursachen haben. Zum einen kann dies durch das Modell beabsichtigt sein, indem alle Marken aus dem Netz entnommen worden sind (z.B. wenn Marken Aufträge darstellen und deren Beendigung den Auftrag aus dem System nimmt). Wenn aber noch Marken vorhanden sind und trotzdem kein Ereignis feuern kann, so liegt ein Deadlock (s.u.) vor. Dieser zeigt entweder einen Fehler im Modell oder in dem Modell zugrundeliegenden System vor. Das Auftreten von Deadlocks und ihre Entstehungsursachen ist daher meist von besonderem Interesse.

3.3.2. Analyse von Petri-Netzen

Bestimmte Markierungen haben definierte Namen, wenn sich im Zusammenhang mit der Schaltregel Probleme ergeben ([Ros83], S. 27ff):

- ***Aktivierung (Schaltbarkeit):*** *dies ist der oben geschilderte Normalfall*
- ***Begegnung:*** *mindestens ein Eingangs- und ein Ausgangszustand sind belegt. Daher kann das Ereignis nicht schalten.*
- ***Verzweigungskonflikt:*** *ein Zustand besitzt mehrere Ausgangsereignisse. Die Auswahl des feuernden Ereignisses ist nicht durch die Schaltregel definiert. Wenn die Ausgangsereignisse mit einer bestimmten relativen Häufigkeit (z.B. abwechselnd) feuern sollen, so muß eine „Entscheidungsregel“ eingebaut werden. Abbildung 5 zeigt das Beispiel des abwechselnden Feuerns zweier Ereignisse.*
- ***Wettbewerbskonflikt:*** *Zwei Ereignisse können feuern und haben mindestens einen Ausgangszustand gemeinsam. Die Schaltregel gibt keine Auskunft darüber, welches der Ereignisse ausgeführt wird. Wenn eine Auswahl aufgrund einer relativen Häufigkeit gewünscht wird, muß analog zum Verzweigungskonflikt ein zusätzlicher Entscheidungsregelkreis eingebaut werden.*
- ***Konfusion:*** *dies entspricht einer veflochtenen Konfliktsituation, bei denen nur eine bestimmte Auswahl und Reihenfolge der Ereignisse dazu führt, daß jede Marke ein Ereignis auslösen kann, während eine andere Auswahl mindestens eine Marke blokkiert.*

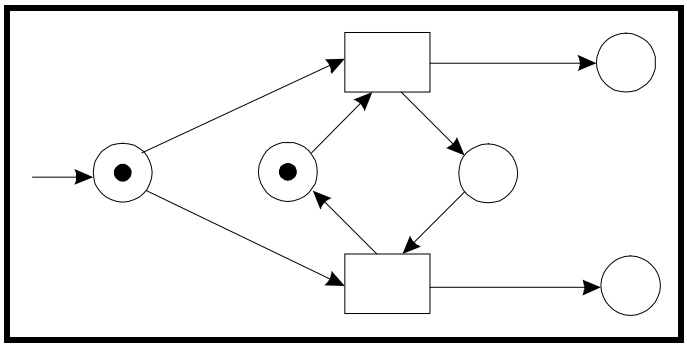


Abbildung 5: Auflösung des Verzweigungskonflikts
(aus [Ros83], S. 28, Abb. 18)

Diese Problemdefinitionen zeigen bereits, für welche Bereiche sich Petri-Netze besonders eignen und genutzt werden. Dies betrifft Zugriff und Verwaltung von gemeinsam nutzbaren Ressourcen und die Synchronisation von parallel ablaufenden Prozessen. Vor allem bei der Verifikation von Netzwerkprotokollen und Betriebssystemroutinen wurden Petri-Netze eingesetzt. [Jen92-1] stellt weitere Beispiele für den Einsatz von Petri-Netzen vor.

Desweiteren lassen sich für Petri-Netze Netztypen definieren, deren Graphen bestimmte Eigenschaften vorweisen:

- **Condition/Event-Netze (C/E-Netz):** Kein Ereignis oder Zustand ist mehr als einmal definiert. Da Ereignisse und Zustände durch ihre Vor- und Nachbedingungen beschrieben werden können, bedeutet das, daß keine zwei Ereignisse oder Zustände die gleichen Ein- und Ausgangsknoten besitzen.
- **Synchronisationsnetze:** Jeder Zustand besitzt genau ein Eingangs- und ein Ausgangsereignis. Ein solches Netz ist frei von Verzweigungskonflikten.
- **Zustandsnetze:** Jedes Ereignis besitzt genau einen Eingangs- und einen Ausgangszustand. Ein solches Netz kann keine Wettbewerbskonflikte beinhalten.
- **Kausalnetze** sind eine besondere Form von Synchronisationsnetzen. In ihnen kann ein Zustand oder Ereignis nur einmal aktiviert werden, d.h. der Graph ist zyklenfrei.

In Petri-Netzen lassen sich dynamische Eigenschaften des Netzes analytisch untersuchen. Im folgenden werden einige dieser Eigenschaften - konservativ, sicher, lebendig und deadlock-frei - vorgestellt ([Ros83], S. 40f):

- *Ein Netz heißt* **sicher bezüglich einer Anfangsmarkierung**, *wenn durch keine zulässige Anwendung der Schaltregel ein Zustand des Netzes mit mehr als einer Marke belegt werden kann. Ansonsten heißt es* **unsicher.**
- *Ein Ereignis heißt* **lebendig bezüglich einer Anfangsmarkierung**, *wenn es nach endlich vielen zulässigen Schaltvorgängen schalten kann, d.h. seine Eingangszustände sind markiert und seine Ausgangszustände frei. Ansonsten heißt das Ereignis* **tot**.
- *Ein Netz heißt* **lebendig bezüglich der Anfangsmarkierung**, *wenn alle Ereignisse lebendig sind.*
- *Ein Netz mit einer gegebenen Anfangsmarkierung besitzt einen* **Deadlock (Systemstillstand)**, *wenn nach endlich vielen Schaltungsvorgängen eine Markierung erreicht werden kann, in der kein Ereignis mehr feuern kann. Wenn es keinen Deadlock geben kann, so ist das Netz* **deadlock-frei**.
- *Ein Netz ist* **konservativ bezüglich einer Anfangsmarkierung**, *wenn in jedem Ereignis die Anzahl der Marken erhalten bleibt. Dies kann direkt am Petri-Netz durch Untersuchung aller Ereigniskanten analysiert werden.*

Zur Analyse von Petri-Netzen muß in der Regel die Menge aller in diesem Netz möglichen Markierungen (ausgehend von einer gegebenen Ausgangsmarkierung) und das Zusammenwirken zwischen den Markierungen untersucht werden. Diese Menge läßt sich als Graph darstellen, den man als ***full occurrence graph (O-graph, oc-Relation)*** bezeichnet. Zur Definition des O-Graphen werden folgende einleitende Definitionen über Markierungen benötigt ([Jen92-1], S. 76f)

- *Die Markierung M_2 ist* **direkt erreichbar** *von der Markierung M_1 aus, wenn in der Markierung M_1 das Ereignis Y feuern kann und die Markierung M_2 zum Ergebnis hat. Schreibweise:* $M_1[Y\rangle M_2$
- *Die Markierung M_{n+1} ist* **erreichbar** *von der Markierung M_1 aus, wenn es eine Folge von Aktivierungen von Ereignissen gibt, so daß die Markierung M_{n+1} erreicht wird. Schreibweise:* $M_1[Y_1\rangle M_2[Y_2\rangle \ldots M_n[Y_n\rangle M_{n+1}$

  Analog läßt sich eine unendliche Aktivierungssequenz definieren.
- *Die Menge aller von einer Markierung M aus erreichbaren Markierungen wird durch* $[M\rangle$ *ausgedrückt.*

Der Occurence-Graph eines Petri-Netzes mit Anfangsmarkierung M_0 ist dann ein gerichteter Graph mit Knoten *V*, Kanten *A* und Knotenfunktion $N: A \to V \times V$ [4], wobei gilt ([Jen92-2], S.5):

- $V = [M_0\rangle$
- $A = \{(M_1, M_2) \in V \times V \mid M_1[Y\rangle M_2\}$
- $\forall a = (M_1, M_2) \in A: N(a) = (M_1, M_2)$

An gleicher Stelle findet man auch einen Algorithmus zur Erstellung von O-Graphen. Durch Analyse des O-Graphen können Aussagen über die Eigenschaften des Petri-Netzes getroffen werden. Das Petri-Netz ist genau dann beschränkt, wenn der O-Graph endlich ist. Die Erreichbarkeit kann durch die Suche nach einem Pfad im O-Graphen bestimmt werden. Ein Deadlock ist durch einen Knoten im O-Graph ohne Ausgangskanten dargestellt. Weitere Analyse-Regeln werden in ([Jen92-2], S. 11-17) beschrieben.

3.3.3. Erweiterungen

Im Laufe der Zeit wurden viele Erweiterungen zu Petri-Netzen vorgeschlagen. Einige der wichtigsten sollen an dieser Stelle vorgestellt werden.

Generalisierte Petri-Netze

In generalisierten Petri-Netzen sind auch mehrere Kanten zwischen zwei Knoten in einer Richtung erlaubt.

Verhinderungskanten

Verhinderungskanten sind Kanten, die das Feuern eines Ereignisses bedingen, wenn im Ausgangszustand keine Marke vorliegt. Durch Verhinderungskanten werden Petri-Netze äquivalent zu Turing-Maschinen ([Pag94], S.64).

[4] Diese Definition erlaubt auch Mehrfachkanten und bezieht sich auf ein Petri-Netz mit Mehrfachkanten.

Mehr-Marken-Netze

In Mehr-Marken-Netzen ist es erlaubt, daß sich in einem Zustand mehr als eine Marke befindet. Zusätzlich wird zu jeder Kante ein Kantenausdruck hinzugefügt, der aussagt, wieviele Marken entfernt bzw. hinzugefügt werden Die Schaltregel wird nun so interpretiert, daß in jedem Eingangszustand mindestens soviel Marken vorhanden sein müssen, wie in dem Kantenausdruck zum Eingangszustand zum Ereignis angegeben werden. Durch das Feuern werden diese Marken aus einem Eingangszustand entfernt und im Ausgangszustand werden so viele Marken hinzugefügt, wie im Kantenausdruck zwischen Ereignis und Ausgangszustand angegeben sind. Anstelle der Sicherheit eines Netzes wird nun eine ähnliche Definition eingeführt. Ein Netz ist ***k-beschränkt***, wenn maximal *k* Marken in jedem Zustand vorliegen können.

Diese drei Erweiterungen sind bereits so stark in das ursprüngliche Konzept integriert worden, daß viele Autoren unter der Bezeichnung „Petri-Netze“ die im vorherigen Abschnitt vorgestellte Version mit diesen Erweiterungen verstehen ([Pag94], S. 64, auch [Jen92-1]). Die folgenden Erweiterungen werden hingegen oft als ***high-level petri-nets*** bezeichnet.

Coloured Petri Nets (CPN)

In der klassischen Petri-Netz-Theorie sind Marken identische, eigenschaftslose Objekte, die in Zuständen vorhanden sein können. Eine Erweiterung der Petri-Netze, die von Kurt Jensen stammt (s. [Jen92-1]), ändert diese Tatsache ab. Marken werden dort als Objekte eines bestimmten Typs definiert. Da in den einfachen Petri-Netze die Marken als kleine schwarze Kreise dargestellt werden, wurde für die Erweiterung der Begriff der „Coloured Petri Nets“ eingeführt, der aussagen soll, daß Marken jetzt eine bestimmte Farbe (colour) aus einer vorgegebenen Menge von Farben (colour set) haben.

Zusätzlich zu der Definition von Farbmengen und Farben für Marken wird jedem Zustand eine Farbe zugewiesen. Diese sagt aus, daß in diesem Zustand sich nur Marken von dieser Farbe befinden können. Durch die Typisierung von Marken kann zwar die generelle Leistungsfähigkeit von Petri-Netzen nicht gesteigert werden, allerdings wird dadurch eine einfachere, kompaktere und verständlichere Modellierung möglich.

Timed Petri Nets und Stochastische Petri-Netze

Petri-Netze enthalten keine Informationen über das zeitliche Verhalten eines Systems. Alle Ereignisse erfolgen zeitlos. Aus diesem Grund wurden mehrere Vorschläge zur Erweiterung von Petri-Netzen vorgeschlagen, welche Zeit mit in das Modell einbringen. Dabei gibt es i.A. zwei Möglichkeiten:

- Zeit kann mit den Ereignissen verknüpft sein. Jedes Ereignis hat eine Ausführungsdauer. Zu Beginn des Feuerns werden dabei die Marken in den Eingangszuständen entfernt. Aber erst nach Beendigung der Ausführungsdauer werden sie in den Ausgangszuständen hinzugefügt.
- Zeit kann mit den Zuständen verknüpft sein. Wenn eine Marke hinzugefügt wird, so können die Ausgangsereignisse erst nach Ablauf einer bestimmten Dauer feuern.

Generell ist auch eine Mischform aus beiden möglich, diese wird aber in der Praxis nicht verwendet, da eine der beiden Methoden zur Modellierung ausreicht und ein klareres Modell liefert. Üblich ist die Verknüpfung mit Ereignissen. Ist die Zeitdauer fest, so spricht man von „Timed Petri Nets“. Können für die Dauer auch zufällige Variable verwendet werden, so werden die so entstehenden Petri-Netze „stochastisch“ genannt (stochastic petri nets, SPN).

3.3.4. Queueing Petri Nets (QPN) und Simulation Nets

Es existieren verschiedene Zusammenfassungen von Petri-Netzen mit einer Zeitkomponente und typisierten Marken, zu dem ein Warteschlangenkonzept hinzugefügt wird. Dadurch erhält man einen For-

malismus, der sich besonders zur Erstellung von Simulationsmodellen eignet. Besonderes Interesse liegt dabei an parallelen Simulationen, da Petri-Netze durch ihr implizit paralleles Verhalten sich für diesen Zweck besonders eignen.

„Simulation Nets“ wurden von Aimo Törn entwickelt und dienen als Grundlage für ein Simulationswerkzeug auf der Basis von Petri-Netzen [Gus94]. Dazu wurden zu bereits erwähnten Netzkomponenten noch Unterbrechungskanten implementiert. Diese können ein bereits gefeuertes, aber noch nicht vollendetes Ereignis wieder zurücknehmen.

„Queueing Petri Nets“ (QPN) [Bau96] versucht die Vorteile von Petri-Netzen und Warteschlangennetzen zusammenzuführen. Dazu werden an den Zuständen Warteschlangen hinzugefügt und das zeitliche Verhalten mit den Zuständen verknüpft. Eine in einem Zustand ankommende Marke wird in die Warteschlange eingereiht und ist nicht an den Ausgangsereignissen abrufbar. Erst nach Beendigung von Wartezeit und Bedienzeit kann die Marke andere Ereignisse aktivieren. Somit wird der Zustand durch ein Warteschlangen-Bedieneinheit-Modell aus der Warteschlangentheorie ersetzt. Auf QPNs können sowohl die qualitativen Analysen der Petri-Netz-Theorie als auch quantitative Methoden der Warteschlangentheorie angewendet werden.

3.3.5. Zusammenfassung

Petri-Netze bieten einige bemerkenswerte Vorzüge, die ihren Einsatz als ein Werkzeug zur Modellierung von Systemen rechtfertigen. Petri-Netze können zur Beschreibung eines bestehenden Systems, zur Spezifikation eines zu implementierenden Systems oder als Präsentation eines Systems verwendet werden. Zur Analyse können sowohl formale Methoden als auch Simulation eingesetzt werden.

Durch die graphische Repräsentation von Zuständen und Ereignissen stellen Petri-Netze eine vollständige Modellbeschreibung dar. Durch die klar definierte Semantik mit einfachen Konstrukten ist das Verhalten eines Petri-Netzes eindeutig. Die Möglichkeiten, hierarchische Netze zu konstruieren, erlauben einen Wechsel der Abstraktionsebenen, durch die man auch große Systeme durch Dekomposition untersuchen kann. Durch die Lokalität der Schaltregel ist ein System relativ stabil gegenüber kleinen Veränderungen, so daß der Modellierer ohne großen Aufwand Parameterstudien oder Optimierungsvorschläge testen kann. Die in Petri-Netzen implizit vorhandene Parallelität erlaubt das einfache Modellieren nebenläufiger Prozesse. Durch die vielen Erweiterungsvorschläge stehen dem Modellierer Methoden zur Verfügung, sich auf das Wesentliche des Systems zu konzentrieren, ohne durch das Konzept behindert zu werden. Die Erstellung, Analyse und Simulation von Petri-Netzen kann sehr gut durch Computerprogramme unterstützt werden. Wegen der formalen Analysemethoden werden Petri-Netze oft zum Entwurf und Verifikation von parallelen Prozessen eingesetzt.

Die Modellierung eines Fertigungssystems mittels Petri-Netze hat sowohl Vor- als auch Nachteile. Die Unterteilung in Zustände und Ereignisse löst das Problem der Ereignisgraphen, die keine explizite Darstellung der Zustände besitzen. Die Möglichkeit, hierarchische Verfeinerungen anzugeben, ist zur übersichtlichen Modellierung eines komplexen Fertigungssystems notwendig. Die Probleme ergeben sich aus der starken Semantik der Petri-Netze. Diese schränken die Modellierung ein, so daß oft umständliche und wenig intuitive Modelle entstehen. Diese Einschränkung ist dann zu akzeptieren, wenn aus den Analysemöglichkeiten für Petri-Netze ein großer Nutzen gezogen werden kann. Dieser ist allerdings für die Simulation eines Fertigungssystems nicht zu erkennen, da hier das Ziel der Optimierung vorliegt. Die Möglichkeiten, gewisse Existenzbeweise (z.B. daß alle Operationen auch tatsächlich bearbeitet werden, oder daß kein Deadlock durch sich gegenseitig blockierende Ressourcen vorliegen kann) zu führen, sind für diese Simulationsanwendung eher uninteressant. [Pag94] kritisiert an dem Konzept der Petri-Netze weiterhin das Fehlen von expliziten Komponenten- und Experimentverwaltungsmechanismen und High-Level-Repräsentationsmöglichkeiten.

3.4. Weitere formale Methoden

Die in den vorangehenden Abschnitten vorgestellten Methoden - DEVS, Petri-Netze und Ereignisgraphen stellen nur einen Ausschnitt aus einer Vielzahl von Methoden dar. Insbesondere konzentrieren sich die betrachteten Methoden auf die diskrete ereignisgesteuerte Simulation. In diesem Abschnitt werden weitere Methoden vorgestellt, die andere Schwerpunkte setzen. Allerdings würde eine detailliertere Betrachtung den Rahmen dieser Arbeit sprengen, so daß die folgenden Methoden nur kurz angerissen werden. Für weitere Informationen zu den einzelnen Methoden wird auf die jeweils angegebene Literatur verwiesen.

3.4.1. Veränderungskalkül (Change Calculus)

Eines der ältesten formalen Modellierungsmethoden ist das Veränderungskalkül von Michael Lackner, wozu [Pag94] eine detaillierte Einführung gibt. In dieser Methode steht die Veränderung von Zustandsvariablen im Vordergrund. Eine Simulation ist in dieser Theorie das Abbilden des zu untersuchenden Systems auf einen Systemzustand und einen Algorithmus, der eine Folge von Systemzuständen produziert, was als Verhalten des Systems interpretiert werden kann. Der Systemzustand wird dabei als *n*-dimensionaler Vektor von Zustandsvariablen betrachtet, wobei *n* im Laufe der Simulation nicht konstant sein muß. Das Verhalten des Modells wird durch eine Menge von Ausdrücken beschrieben. Eine einzelne Veränderungsrelation wird durch $\alpha:\beta$ beschrieben, was bedeutet, wenn α erfüllt ist, dann gilt β. α und β sind Konjunktionen oder Zustandsbeschreibungsvariablen. α heißt Vorgänger, β Konsequenz. Konjunktionen können mittels weniger Primitiver aus den Zustandsvariablen erstellt werden. Dazu gibt es eine Menge von Regeln, die angeben, unter welchen Umstäden α durch β ersetzt werden kann, was die Veränderung des Systems beschreibt. Ein System wird durch eine Menge von Veränderungsrelationen und einen Ausgangszustand beschrieben, auf den dann schrittweise eine der gültigen Übergangsregeln angewendet werden kann.

Das Veränderungskalkül hat sich aber seit seiner Veröffentlichung in den sechziger Jahren nicht durchsetzen können, was an der kryptischen und schwierigen Schreibweise und fehlenden graphischen Modellierungsmethoden liegt. Weiterhin kann man das Veränderungskalkül durch Umformungen auf einen Endlichen Automaten reduzieren.

3.4.2. Activity Cycle Diagrams (ACD)

Eine weitere Methode aus den sechziger Jahren sind die Activity Cycle Diagrams (ACD) von K. D. Tocher, die in [Pag94] beschrieben werden. ACD stellen den logischen Fluß einer Simulation auf graphische Weise dar. Im Prinzip sind sie ähnlich zu Flußdiagrammen, die den logischen Fluß eines Computerprogramms aufzeigen. In einem ACD-basierten Modell wird ein Simulationsmodell durch einen Menge von ***Entitäten*** beschrieben, die miteinander in Verbindung stehen. Eine Entität kann entweder ***ruhend*** oder ***aktiv*** sein. Die Aktivität einer Entität steht in der Regel in Zusammenhang mit anderen Entitäten. Die Dauer einer Aktivität kann im voraus bestimmt werden, wohingegen die Dauer des ruhenden Zustands - auch „queue“ genannt - nicht festgelegt ist, da die Entität auf eine Aktivität einer anderen Entität wartet. Aktivitäten werden in einem ACD durch Rechtecke, Wartephasen durch einen Kreis dargestellt. Konventionsgemäß folgt auf einen Aktivität immer eine Ruhephase und umgekehrt, was dazu führt, daß Leerphasen eingebaut werden müssen. Außerdem müssen Kreisläufe in ACD geschlossen sein. Das in Abbildung 6 dargestellte Beispiel modelliert einen englischen Pub, der aus drei Entitäten besteht: einem Mann, einer Bedienung und einem Glas ([Pag94], S. 51). Der Mann trinkt aus seinem Glas oder wartet, die Kellnerin tut nichts oder schenkt das Glas wieder voll, das Glas ist entweder leer oder voll, wird gerade nachgeschenkt oder ausgetrunken.

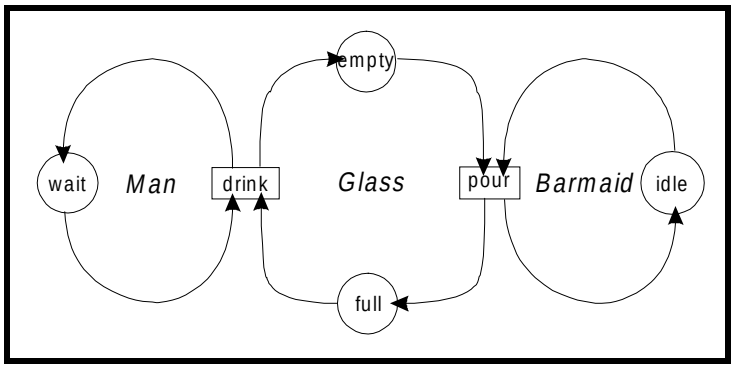


Abbildung 6: Activity Cycle Diagram für das English Pub Model (aus [Pag94], S. 52)

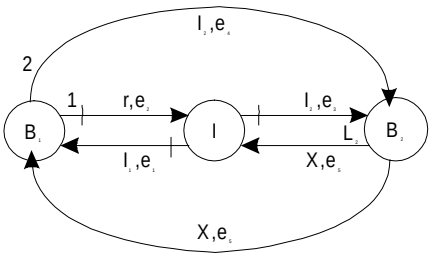
ACD können als eine vereinfachte Variante von Petri-Netzen betrachtet werden und zeigen ähnliche Schwächen und Stärken. Die Stärken liegen in der Synchronisation paralleler Prozesse (wie dem Mann und der Bedienung aus dem obigen Beispiel). Die Schwächen von ACDs sind denen von einfachen Petri-Netzen vergleichbar, es fehlen Erweiterungen für Zufälligkeiten und Dauer von Aktivitäten.

3.4.3. Kontrollflußgraphen (Control Flow Graphs)

Kontrollflußgraphen (Control Flow Graphs, CFG) sind eine Methode, die sich besonders für parallele Modelle eignen. Entwickelt wurden sie von Cota und Sargent als ein Mittel, um parallele Simulationsalgorithmen zu entwickeln, sie können aber auch sehr gut auf die Modellierung angewendet werden.

In einem CFG werden in einem gerichteten Graph das Verhalten eines Prozesses oder einer Prozessklasse beschrieben. Jeder Knoten des Graphen stellt einen Zustand des Prozesses dar, den man als ***Kontrollzustand*** oder ***Synchronisationspunkt*** bezeichnet. In diesem Zustand wartet der Prozess darauf, daß eine bestimmte Bedingung erfüllt wird oder eine definierte Zeitdauer abgelaufen ist. Jede Aktion, die der Prozess dann ausführen kann, wird durch eine Kante dargestellt. Verschiedene Prozesse kommunizieren durch ***Botschaften (message passing)*** miteinander. Jeder Prozess besitzt eine Menge von Zustandsvariable zur Beschreibung des Prozesszustands, Eingabe- und Ausgabekanäle zur Übermittlung der Botschaften. Mit jedem Eingabekanal ist eine Warteschlange von noch nicht bearbeiteten Botschaften verbunden. Der empfangende Prozess kann entscheiden, wann er die nächste Meldung empfängt (d.h. aus der Warteschlange abholt), muß sich aber an die Reihenfolge des Eintreffens halten. Botschaften übernehmen hier die Rolle von Ereignissen. Jede Kante eines Zustandsknoten ist mit einem Botschaftstyp, einer Bedingung und einer Kantenpriorität verbunden. Der Botschaftstyp bestimmt, ob eine Veränderung des Prozesszustands oder das Verschicken einer anderen Botschaft ausgelöst wird. Dazu muß die Bedingung der Kante erfüllt sein. Sind die Bedingungen mehrerer Kanten erfüllt, so entscheidet die Priorität darüber, welche Kante ausgewählt wird. Abbildung 7 zeigt eine präemptive Bedienstation mit einer Warteschlange, d.h. höher priorisierte Anforderungen unterbrechen eine gerade ablaufende Bedienung.

[Pag94] kritisiert an CFG, daß die Konstruktion des Graphen zu kompliziert ist und zu viele Details nur implizit repräsentiert werden können. Dort findet man auch eine detaillierte Einführung und weitere Literaturhinweise.



Control states	
I	idle
B_1 (B_2)	busy, serving class 1 (2) transaction
Input channels	
I_1 (I_2)	Input channel for class 1(2) transaction
Output channels	
O_1 (O_2)	Output channel for class 1 (2) transaction
State variable	
m_1 (m_2)	Class 1 (2) transaction in service
r	service time of a class 1 transaction
s	time at which service on a class 1 transaction begins
Eventtypes	
e_1	{begin service of class 1 transaction} receive m_1 from I_1 compute r; set s to current clock
e_2	{complete service of class 1 transaction} send m1 to O_1 set m1 to null
e_3	{begin service of class 2 transaction} receive m_2 from I_2
e_4	{preempt class 1 transaction} set r=r-(current clock -s) receive m_2 from I_2

Abbildung 7: Kontrollflußgraph für Bedieneinheit mit Unterbrechung (aus [Pag94], S. 72f)

3.4.4. Markovsche Prozesse

Die Anwendung der Theorie der Markovschen Prozesse zur Modellierung von Systemen geht weiter zurück als der Einsatz von computerbasierten Simulationsmodellen. Der Hauptgesichtspunkt liegt dabei in der theoretischen Analyse aufgrund stochastischer Gesetzmäßigkeiten als Alternative zur Simulation. Das zeitabhängige Verhalten eines Systems wird dabei durch eine zufällige Variable X beschrieben, die zu Zeitpunkten t_i beobachtet wird. Das Verhalten von X kann durch das Tupel $\{X(t_i), t_i\}, i = 1,2,\ldots$ dargestellt werden. Ein ***stochastischer Prozess*** ist Abbildung dieser Tupel auf eine Familie von Zufallsvariablen $\{X(t), t\}, X(t) \in \Omega, t \in \Gamma$. Ω ist dabei der Zustandsraum von X, Γ die Indexmenge (Parametermenge, oder kurz „Zeit“). Ω und Γ können dabei diskret oder kontinuierlich sein, so daß sich insgesamt vier Klassen von stochastischen Prozessen ergeben: zustandsdiskrete bzw. zustandskontinuierliche, zeitdiskrete bzw. zeitkontinuierliche. Sehr häufig werden zustandsdiskrete, zeitkontinuierliche Prozesse, sog. „Ketten“, verwendet. Ein wichtiges Kriterium für einen stochastischen Prozess ist die ***Markov-Eigenschaft (Gedächtnislosigkeit)***:

$$P\left(X(t_{n+1}) = x_{n+1} \middle| X(t_n) = x_n, \ldots, X(t_0) = x_0\right) = P\left(X(t_{n+1}) = x_{n+1} \middle| X(t_n) = x_n\right), t_0 < t_1 < \ldots t_{n+1}$$

Das bedeutet, daß die zukünftige Entwicklung des Prozesses nur vom aktuellen Zustand x_n abhängt. Auf welchem Weg der Zustand x_n erreicht wurde, ist unerheblich. Ein stochastischer Prozess mit dieser Eigenschaft wird ***Markov-Prozess*** genannt. Eine weitere wichtige Eigenschaft von stochastischen Prozessen ist die ***Erneuerungseigenschaft***. Ein Erneuerungsprozess ist eine abzählbare Menge von reellwertigen Zeitpunkten, bei denen die Abstände zwischen zwei benachbarten Zeitpunkten durch unabhängige und identische Verteilungsfunktionen beschrieben werden können. Warteschlangenmodelle können durch solche Prozesse beschrieben werden und analytisch auf Eigenschaften wie mittlere Wartezeit, Durchlaufzeit, mittlere Anzahl wartender Einheiten, Auslastung, Wartewahrscheinlichkeiten, usw. untersucht werden. Allerdings wird diese Analyse bei komplexen Warteschlangensystemen (z.B. Netzen) sehr schwierig bis unmöglich. In Abschnitt 7.1 wird bei der Evaluation des implementierten Systems ein

einfaches Warteschlangensystem vorgestellt, das auch analytisch untersucht werden kann. Eine Einführung in die Leistungsbewertung von Warteschlangenmodellen und ihren Grundlagen gibt [Tra96].

Eine Erweiterung von Markov-Prozessen, die sich sowohl analytisch als auch durch Simulation untersuchen lassen, sind ***Verallgemeinerte Semi-Markov-Prozesse (generalized semi-Markov process, GSMP)***, die in ([Pag94], S. 75ff) vorgestellt werden. Die in Markov-Prozessen definierten Prozesszeitpunkte werden dabei als Ereignisse interpretiert. Die Verteilung der Ereignisse kann dabei vom Ereignis, dem aktuellen und vorherigen Zuständen und dem Ereignis, das den letzten Zustandsübergang verursachte, abhängen. Durch ein Ereignis kann ein GSMP in einen anderen Zustand wechseln, in dem es dann für eine bestimmte, von diesem Ereignis abhängige Dauer verbleibt. Die Zustands- und Ereignismengen sind diskret und endlich.

Durch GSMP können somit auf einer theoretisch fundierten Basis Prozesse zur Beschreibung des Verhaltens eines Systems definiert werden. In [Pag94] findet sich der Hinweis, daß GSMP und stochastische Petri-Netze äquivalent sind, was zeigt, daß GSMP ausreichend mächtig sind. Allerdings fehlen außer der aus der Warteschlangentheorie bekannten Systemdarstellung weitere, mächtigere graphische Eingabemöglichkeiten. Viele Simulationssprachen haben allerdings Konzepte aus der Warteschlangenmodelle (Ankunftsprozess, Warteschlange, Bedienprozess) übernommen, so daß sich dort erstellte Modelle auch gut mit den analytischen Methoden untersuchen lassen, was eine Art Doppelnutzung des Modells für mehrere Methoden bedeutet.

3.4.5. Differentialgleichungen und Differenzengleichungen

Auf kontinuierliche Zustandsräume sind viele der bisher betrachteten Methoden nicht anwendbar. Eine für diese Zustände übliche Methode ist, das Verhalten dieser Variablen über eine Menge von Differentialgleichungen, welche die Veränderung der Variablen über die Zeit angeben, zu definieren. Kontinuierliche Zustandsvariable findet man oft in der Beschreibung physikalischer Prozesse, z.B. für die Wärmeleitkapazität von Materialien. Allerdings ist nur für wenige Differentialgleichungen eine analytische Lösung möglich. Die Simulation eines solchen Gleichungssystems wird in der Regel dadurch durchgeführt, daß die Differentialgleichungen in Differenzengleichungen umgeformt werden. Diese werden dann ausgehend von einen Anfangswert mit einer kleinen Schrittweite Δt schrittweise berechnet, wodurch sich ein Verlauf der Zustandsvariablen über die Zeit ergibt. Die Wahl einer genügend kleinen Schrittweite ist allerdings i.a. schwierig, da sich sowohl durch zu große Schrittweiten Abweichungen der Differenzengleichung von der ursprünglichen Differentialgleichung als auch bei zu kleinen Werten numerische Rundungsfehler und Effizienzverluste ergeben. Auch die Umwandlungen von Differenzengleichungen in Differentialgleichungen ist üblich, wenn es sich um große diskrete Wertebereiche handelt. Diese Technik wird oft in geographischen Modellen, z.B. zur Beschreibung des Bevölkerungswachstums, verwendet.

Als Beispiel soll das Gleichungssystem eines Räuber-Beute-Systems vorgestellt werden, das in der Biologie häufig verwendet wird ([Fer94], S. 14):

- $\frac{dN_1}{dt} = r_1 N_1 - P N_1 N_2$, $\frac{dN_2}{dt} = a P N_1 N_2 - d_2 N_2$, *mit*
- **P** *der Raubkoeffizient,*
- $\mathbf{N_1}$ *die Population der Beute,*
- $\mathbf{N_2}$ *die Population der Räuber,*
- **a** *Effizienz der Räuber beim Umwandeln von Nahrung in Nachkommen,*
- $\mathbf{r_1}$ *Geburtsrate der Beute,*
- $\mathbf{d_2}$ *Sterberate der Räuber*

Bei der Untersuchung von Differential- bzw. Differenzengleichungen ist die Frage nach einem stabilen Endzustand (steady state) und der Wiederholung von Mustern, sowohl periodischer als auch frak-

taler Natur, interessant. Verwendet werden Differentialgleichungen v.a. zur Beschreibung von physikalischen, biologischen oder geographischen Systemen.

3.4.6. Zelluläre Automaten

Zelluläre Automaten (Cellular Automata, CA) sind eine Modellierungsmethode für Systeme, die aus einer Menge von gleichartigen Einzelteilen, genannt Zellen, bestehen. Die Zellen sind dabei in einem regelmäßigen Muster angeordnet. Jede Zelle ist durch einen Zustand aus einer endlichen Zustandsmenge charakterisiert. Das Verhalten einer Zelle ist nur vom Zustand der Zelle und von seinen „Nachbarzellen" abhängig. Alle Zellen sind in ihrem Verhaltensmuster gleich, d.h. zwei Zellen mit dem gleichen Zustand und denselben Eingaben von ihren jeweiligen Nachbarn reagieren in derselben Weise. Die Veränderungen der Zellen erfolgen gleichzeitig durch eine getaktete Zeitfortschreibung. Die Nachbarschaftsrelation ist lokal und für alle Zellen gleich. Nach [Wei96] kann ein Zellulärer Automat formal durch das Tupel *(L, S, N, f)* definiert werden, wobei

- **L** *ein reguläres Gitter (deren Elemente Zellen heißen),*
- **S** *eine endliche Menge von Zuständen,*
- **N** *eine endliche Menge der Mächtigkeit n von Nachbarschaftsindizes, so daß* $\forall c \in N, \forall r \in L: r + c \in L$ *und*
- $f: S^n \to S$ *eine Zustandsübergangsfunktion ist.*

Ein Beispiel für ein reguläres Gitter ist $\boldsymbol{Z}^n$. In der Regel wird für L ein 2- oder 3-dimensionaler (endlicher) Array gewählt. Eine Konfiguration eines CA ist eine Funktion $C_t: L \to S$, die jeder Zelle einen Zustand zuweist. Der Effekt der Übergangsfunktion f ist das Überführen einer Konfiguration C_t in die Konfiguration C_{t+1} durch $C_{t+1}(r) = f\left(\{C_t(i): i \in N(r)\}\right)$, wobei $N(r) = \{i \in L: r - i \in N\}$ die Menge der Nachbarn der Zelle r ist.

Das einfachste und bekannteste Beispiel aus dem Bereich der CA ist das „Game of Life" (s. Abbildung 8), das die Ausbreitung einer Zellkultur über eine Fläche zeigt. In diesem Beispiel kann eine Zelle nur zwei Zustände, leer oder belebt, haben. Die Übergangsregeln geben an, wann ein freies Feld besiedelt werden kann und wann eine besiedelte Zelle abstirbt, da sie von den umgebenden besiedelten Zellen zuwenig Nahrung (Licht, Luft) bekommt.

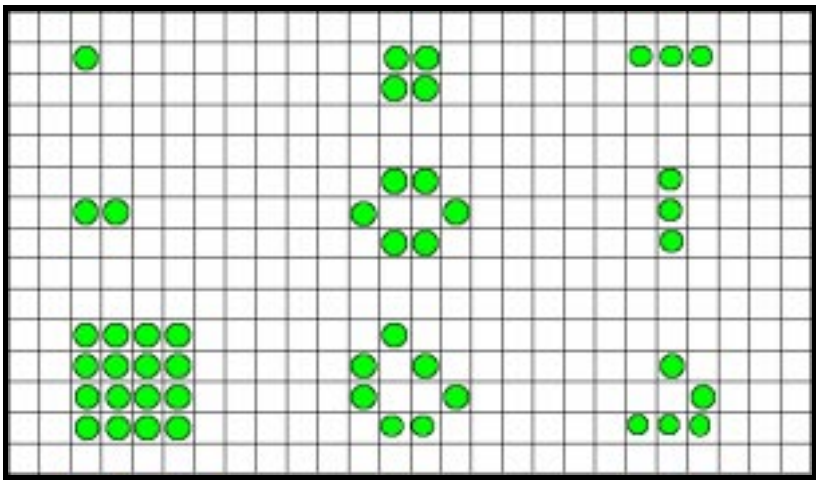


Abbildung 8: John Convay's „Game of Life" (aus [Wei96], Fig. 1)

Zelluläre Automaten sind sehr einfach zu modellieren und implementieren, zeigen aber sehr häufig ermergente Verhaltensweisen durch das Bilden von Mustern, Selbstähnlichkeiten, Perioden, u.ä.. Die häufigsten Einsatzgebiete von CA liegen in den Bereichen Artificial Life und der Modellierung von biologischen, medizinischen oder geographischen Systemen. In [Vea94] wird die Simulation der Ausbrei-

tung von Waldbränden über eine Landschaft vorgestellt, die mit CA implementiert wurde. Eine detaillierte Einführung in Zelluläre Automaten und weitere Literaturhinweise finden sich in [Wei96].

3.4.7. Agenten

Eine Modellierungsmethode mit einem anderen Ansatz ist die Modellierung von Agentensystemen, die aus den Gebieten der Künstlichen Intelligenz und Artificial Life stammt. Dort wird ein ***Agent*** als ein System, „das sich in einer Umgebung befindet und Teil dieser Umgebung ist, auf die Umgebung im Laufe der Zeit einwirkt und so das beeinflußt, was es in der Zukunft wahrnehmen kann“ ([Klu97], S. 2, übersetzt) definiert. Während auch in anderen Modellen die Vorstellung von einzelnen agierenden Einheiten (z.B. Prozesse in CFG, entities in ACD) existiert, kommt hier der Idee der ***Umgebung*** eine wichtige Rolle zu.

Nach seinen Eigenschaften läßt sich ein Agent in verschiedene Kategorien - reaktiv, autonom, zielorientiert, kommunikativ, lernend, mobil, flexibel, charakterbasiert - einteilen. Die in Simulationen am häufigsten verwendete Gruppe sind reaktive Agenten. Diese besitzen kein Wissen über ihre Umgebung, sondern handeln ohne große Verzögerung auf Ereignisse ihrer Umwelt. Die Reaktion erfolgt in der Regel „gedächtnislos“, d.h. ein Agent benötigt kein Gedächtnis über seine Handlungen. Komplexere Agentenarchitekturen sind „kognitive Agenten“, die Wissen über ihre Umwelt erwerben und verarbeiten können, wozu sie eine Repräsentation der Umwelt und der anderen Agenten benötigen. Eine weitere, sich mit den „kognitiven Agenten“ überschneidende Architektur stellen „Intentional agents“ dar, deren innerer Zustand durch Ausdrücke wie „beliefs“, „intentions“, „desires“, usw. beschrieben werden kann. Ein Spezialfall dieser Architektur sind BDI-Agenten, die über drei intentionale Komponenten „Beliefs“ (das eigene Modell der Umwelt und anderer Agenten), „Desires“ (langfristige Ziele) und „Intentions“ (kurzfristige Pläne) verfügen. Durch die Modellierung der Agenten und ihrer Umwelt kann eine Simulation des Modells aufgrund des Verhaltens der einzelnen Agenten durchgeführt werden, was als „Multi-Agenten-Simulation“ (many-agent simulation) bezeichnet wird. Eine Multi-Agenten-Simulation besteht aus vier Komponenten ([Fer94], S. 15):

- **agents**, *die Menge aller Agenten*
- **objects**, *die Menge aller passiven Entitäten*
- **environment**, *der Raum, in dem sich Agenten und Objekte befinden, bewegen, kommunizieren, usw.*
- **communications**, *die Menge aller Kommunikationsmittel*

Die Simulation läuft so ab, daß nach Initialisierung der Umwelt und der Agenten ein Agent ausgewählt wird, der seine Eingaben (Ereignisse, Wahrnehmungen) untersucht und aufgrund dessen (und der Kenntnis seiner Umwelt, falls er eine besitzt) seine Aktionen ausführt. Danach wird der nächste Agent ausgewählt, usw. Das Fortschreiten der Zeit kann entweder getaktet, wobei dabei in jedem Schritt alle Agenten der Reihe nach ausgewählt werden, oder ereignisorientiert erfolgen. Beim ereignisorientierten Ansatz wird immer nur der Agent ausgewählt wird, den dieses Ereignis betrifft. Einen ereignisorientierten Ansatz findet man v.a. in Systemen, in denen die Umwelt selbst wie ein Agent funktioniert (und mit den Agenten „kommuniziert“), die gleichmäßige Zeitfortschreibung in Systemen, in denen die Umwelt als „Landkarte“ von einzelnen Räumen mit verschiedenen Eigenschaften, ähnlich zu Zellulären Automaten, betrachtet wird. Die Ziele agentenbasierter Simulationen dienen dem Test von Hypothesen über die Emergenz sozialer Strukturen aus dem Verhalten der Individuen heraus, dem Erstellen neuer Theorien zum Verständnis des Verhaltens von Systemen und der Integration von Teiltheorien aus verschiedenen Bereichen zu einem großen Gesamtwerk ([Fer94], S. 16). Eine numerische Analyse wird oftmals über den Vergleich des Systemverhaltens mit Meßdaten aus realen Systemen durchgeführt.

Die grundsätzliche Architektur von Agenten als eigenständige Einheiten, die mit anderen kommunizieren, hat viele Ähnlichkeiten mit dem objektorientierten Programmierparadigma, so daß sich eine Implementierung von Agentensystemen auf dieser Grundlage sich anbietet. Die Anwendungen von agen-

tenbasierten Simulationen konzentrieren sich hauptsächlich auf den Bereich biologischer oder soziologischer Systeme, z.B. einer Ameisenkolonie ([Fer94], S.16). Eine ausführlichere Einleitung in agentenbasierte Simulationen findet sich in [Klu97], weitere Literaturhinweise findet man ebenfalls dort und in [Fer94].

3.5. Mischformen

Viele der in diesem Kapitel vorgestellten Methoden können miteinander verbunden werden. Vor allem Methoden, die einen großen Wert auf den hierarchischen Aufbau von Teilmodellen legen, bieten hier viele Vorteile. Die Idee der Gliederung in Teilsysteme ist am stärksten in DEVS ausgeprägt, das keine Aussagen über die Modellierung der atomaren Modelle macht, außer daß sie sich an vorgegebene Kommunikationsformen halten müssen. Dort kann im Prinzip jede der anderen vorstellten Methoden eingebaut werden. Auch Petri-Netze bieten gute Möglichkeiten zur Verfeinerung von Modellen. Dabei kann aufgrund des rein lokalen Schaltverhaltens eine Verfeinerung auch in einer anderen Methoden implementiert werden, solange sie sich an die Ein- und Ausgangsknoten des Netzes anpaßt, d.h. sie muß für einen Transport der Marken sorgen. In Agentensystemen ist es ebenfalls üblich, einfache, reaktive Agenten durch Endliche oder Zelluläre Automaten darzustellen.

Sehr häufig finden sich auch Mischformen von diskreten Methoden mit Differentialgleichungen. Dies ist oft für Systeme notwendig, die sowohl über diskrete als auch über kontinuierliche Zustandsvariablen verfügen. Dabei werden die kontinuierlichen Variablen meistens durch Differenzengleichungen beschrieben und können so recht einfach in eine diskrete Methode eingebaut werden.

3.6. Zusammenfassung

Die vorgestellte Vielzahl von Modellierungsmethoden zeigt an, daß es für das Modellieren eines Systems keine allgemeingültige beste Wahl geben kann. Jede der vorgestellten Methoden hat Stärken und Schwächen, die in jedem Einzelfall sorgfältig gegeneinander abgewogen werden müssen. Insbesondere sind das Ziel einer Simulationsstudie und die Eigenschaften des zu modellierenden Systems bei der Auswahl einer Methode zu beachten. Die Auswahl der vorgestellten Modellierungsmethoden wurde in diesem Abschnitt auf formale Methoden, die unabhängig von der Implementierung sind, beschränkt. Im Abschnitt 5 sollen aus dem ebenfalls großen Angebot der implementationsabhängigen Simulationssprachen einige Beispiele vorgestellt werden. Auch diese sind bei der Wahl der Methode zu berücksichtigen, v.a. wenn ein großer Wert auf die Effizienz und die statistische Aussagekraft gelegt wird, denn Simulationssprachen sind i.a. auf diese Bereiche hin optimiert.

Bei der Modellierung einer Fertigung muß berücksichtigt werden, daß ein Fertigungssystem aus einer großen Anzahl einzelner Komponenten besteht, die parallel einzelne Bearbeitungsschritte durchführen. Für die Durchführung eines Bearbeitungsschritts werden jeweils mehrere Komponenten (z.B. Maschine, Material, Werker) benötigt. Wird das Fertigungssystem auf einem hohen Abstraktionsgrad beschrieben, so ergeben sich für die einzelnen Kompontenen recht einfache Verhaltensmuster wie das Durchführen eines Bearbeitungsschritts auf einem Arbeitsplatz. Dabei können für jede Komponente relativ wenige Zustände wie "arbeitet", "ruht" oder "gestört" definiert werden. Aus diesen Eigenschaften kann für eine geeignete Modellierungsmethode die Forderung gestellt werden, daß sie ein hierarchisches Modellieren zur Gliederung der vielen Komponenten ermöglichen muß. Für die Modellierung der einzelnen Komponenten kann eine relativ einfache Methode verwendet werden, die aber sowohl Ereignisse als auch Zustände der Komponente darstellen soll. Ebenso ist ein Konzept zur Berücksichtigung der Parallelität nützlich. Für die Anwendung analytischer Aussagen über das Modell gibt es bei der Modellierung einer Fertigung wenig Möglichkeiten.

Von den vorgestellten Methoden kann allerdings keine alle Anforderungen erfüllen. DEVS, Ereignisgraphen und Petri-Netze bieten jeweils einen Aspekt an, während aus den anderen Methoden zwar interessante Ansätze übernommen werden können, diese Methoden aber im wesentlichen nicht zur Modellierung einer Fertigung herangezogen werden können. So ist beispielsweise ein agentbasierter Ansatz

durchaus möglich, wäre aber aufgrund der recht einfachen Verhaltensmuster der einzelnen Komponenten übertrieben. In Abschnitt 6.3 wird die Architektur der implementierten Simulationsshell erläutert, die Elemente aus mehreren der vorgestellten Methoden miteinander verbindet.

4. Produktionsprozess und Ressourcenbelegungspläne

4.1. Übersicht

Als ein Teil der Unternehmensplanung wird die Produktionsplanung seit über 30 Jahren durch Rechnersysteme unterstützt. Produktionsplanungs- und -steuerungssysteme (PPS-Systeme) bestehen zum einen aus relativ einfachen, aber umfangreichen Aufgaben (z.B. Auftragsverwaltung, Lagerverwaltung, Bestellschreibung), die effiziente und mächtige Datenbanksysteme benötigen. Diese sind auf heutigen Rechnersystemen möglich, so daß für diesen Aufgabenbereich ausreichend viele Lösungen zur Verfügung stehen. Die angebotenen Systeme bieten je nach Größe, Kosten und Architektur unterschiedliche Möglichkeiten und Ergebnisse. Die Hauptschwierigkeit liegt heute in der Integration verschiedener Datenmodelle und Verwaltungssysteme, die von einzelnen Abteilungen verwendet werden (Stichwort: CIM), dem Management komplexer Softwareprodukte sowie in der Gewinnung von Informationen aus den im Laufe der Zeit anfallenden Datenmengen (Stichwort: Data Mining).

Schwieriger erweist sich die Durchführung der Disposition, wozu die Beschaffungsrechnung, Kapazitätsbedarfsrechnung, Kapazitätsabstimmung und Reihenfolgeplanung gehören. Dabei wird an moderne PPS-Systeme die Forderung gestellt, daß die Lösung dieser Einzelprobleme nicht getrennt, sondern simultan erfolgt. [Ker94] stellt fünf Anforderungen an ein PPS-System:

- **Simultane Planung aller Elementarfaktoren:** Eine sukzessive Planung einzelner Faktoren führt dazu, daß für die zuerst geplanten Faktoren im Überfluß geplant wird und für die zuletzt geplanten Faktoren Sonderfälle und Notlösungen entstehen, was zu wirtschaftlichen Verlusten führt.
- **Grundprinzip der knappen Mengenplanung:** Jeder Elementfaktor muß zwar zur richtigen Zeit in ausreichender Menge vorhanden sein, aber eine zu große oder zu frühe Bereitstellung eines Faktors verursacht Kosten und bindet Kapital.
- **Grundprinzip der knappen Terminplanung:** Für die geplanten Termine müssen Sicherheitsspielräume eingebaut werden. Eine zu großzügige Bemessung dieser Sicherheiten ist ineffizient und erhöht Durchlaufzeiten und Lagerkosten.
- **Echtzeitplanung:** Die Veränderung von Planungsdaten muß möglichst schnell auf die Datenbanken umgesetzt werden und die sich daraus ergebenden Veränderungen müssen in einem neuen Plan berücksichtigt werden.
- **Abteilungsübergreifende Planung:** Anstatt einen Kundenauftrag sequentiell durch die einzelnen Abteilungen des Betriebs zu reichen und jeweils einen Teilbereich dort zu bearbeiten, führt eine Parallelisierung des Planungsprozesses zu einer drastischen Verkürzung der Liege- und Wartezeiten eines Auftrags.

Zumindest die ersten vier Anforderungen betreffen direkt die Komplexität der Dispositionsalgorithmen. Die simultane Planung verbietet die Aufteilung in einfachere Teilprobleme, die knappe Mengen- und Terminplanung stellen weitere Forderungen an die Qualität der Lösung und die Echtzeitplanung benötigt eine hohe Effizienz der Verfahren. Dazu kommt ein hohes Datenvolumen, so daß „brute-force"-Methoden nicht anwendbar sind.

Im Rest dieses Kapitels wird eine Einführung in die Disposition gegeben (Abschnitt 4.2), wobei sowohl die zur Belegungsplanerstellung notwendigen Datenstrukturen als auch Planungsansätze aus verschiedenen Bereichen vorgestellt werden. Es folgt in Abschnitt 4.2.3 eine Erläuterung, welche Einsatzgebiete für Simulation in der Fertigung existieren und welche dieser Bereiche im Schwerpunkt dieser Arbeit liegen. Im Abschnitt 4.4 werden statische Bewertungskriterien für Belegungspläne vorgestellt. Die Bewertungen können allerdings die Zufälligkeiten, denen der Produktionsprozeß unterliegt, nicht beurteilen. Daher werden in Abschnitt 4.5 anhand von Beispielen die vielfältigen Ursachen für zufällige Schwankungen des Fertigungsprozesses erläutert. Diese Zufallsabhängigkeit kann durch den Einsatz dynamischer Bewertungskriterien, die den zeitlichen Verlauf der Umsetzung eines Belegungsplanes

berücksichtigen, besser erfasst werden. In Abschnitt 4.6 werden Ansätze für dynamische Bewertungskriterien vorgestellt.

4.2. Einführung in Ressourcenbelegungspläne

Die zur Disposition benötigten Planungen lassen sich in drei Bereiche einteilen, die unterschiedliche zeitliche Größenordnungen und Abstraktionsebenen besitzen:

- Die **Grobplanung** bewegt sich in der zeitlichen Auflösung von mehreren Monaten und benötigt Prognosen über das Verhalten der Märkte. In der Grobplanung werden grundlegende, langfristige Kapazitätsprobleme wie z.B. die Anschaffung von Maschinen und die Planung von Standorten gelöst.
- In der **Mittelfristdisposition** wird über einen Zeitraum von Wochen über die Annahme von Aufträgen, die Planung von Arbeitsfolgen auf Maschinengruppen und die Bestimmung von Materialbedarfsterminen entschieden.
- Die **Kurzfristplanung (Fertigungssteuerung)** bestimmt die Reihenfolge von einzelnen Arbeitsgängen unter Berücksichtigung aller simultan benötigter Ressourcen für den Bereich einiger Tage.

Unter einem ***Ressourcenbelegungsplan*** ist das Ergebnis einer Kurzfristplanung zu verstehen. Für den Rest dieses Kapitel stehen Ressourcenbelegungspläne im Vordergrund. Synonym werden daher die Bezeichnungen ***Plan*** und ***Belegungsplan*** im Sinne von Ressourcenbelegungsplan verwendet, obwohl diese Begriffe i.a. ein größeres Feld abdecken. Für Vorgehensweisen für die Grobplanung und Mittelfristdisposition, insbesondere zur Bedarfsvorhersage und Bestandsführung, wird auf [Ker94] und [Kur93] verwiesen.

4.2.1. Datenstrukturen für Ressourcenbelegungspläne

In diesem Abschnitt werden die zur Erstellung von Ressourcenbelegungsplänen benötigten Datenstrukturen vorgestellt. Diese sind in PPS-Systemen in der Regel in relationalen Datenbanken abgelegt. Nach der Bezeichnungsweise von [Kur93] sind die wichtigsten Grunddaten eines PPS-Systems:

- Teile
- Erzeugnisstrukturen
- Arbeitsgänge
- Arbeitspläne
- Betriebsmittel-/Arbeitsplätze
- Fertigungsstrukturen

Unter dem Begriff ***Teil (Material)*** werden dabei sowohl Einzelteile, Roh- und Verbrauchsmaterialien als auch Baugruppen und Endprodukte zusammengefasst. Diese Vereinheitlichung erleichtert die Speicherung in einer Datenbank. Für jedes Teil werden dabei in der Teilestamm-Datenbank ca. 50-100 Attribute abgelegt, so daß diese Relation sehr umfangreich ist. Die wichtigsten Attribute betreffen dabei die Identifikation von Teilen sowie Daten über Konstruktion, Disposition, Bedarf, Bestand, Beschaffung, Produktion, Absatz und Kalkulation. Diese Daten werden für eine Vielzahl von Zwecken innerhalb eines Betriebs verwendet, z.B. für Lagerhaltung, Rechnungswesen, Entwicklung und Produktion.

In ***Erzeugnisstrukturdaten*** wird beschrieben, wie ein Produkt als ein Teil aus einzelnen Baugruppen, die wiederum aus Baugruppen oder Einzelteilen bestehen, zusammengesetzt ist. Dabei ist sowohl die Art der Unterteile als auch deren Mengenangabe wichtig. Die Erzeugnisstrukturdaten für ein Produkt lassen sich durch einen gerichteten azyklischen Graphen, der die Relation „geht ein“ bzw. „besteht aus“ für die Bestandteile eines Produkts darstellt, beschreiben. Das Endprodukt ist der Wurzelknoten des Graphen. Für mehrere Produkte, die u.U. als ähnliche Varianten eines Produkts Baugruppen mitein-

ander gemeinsam haben, entsteht dadurch ein ***Gozinto-Graph***[5]. Abbildung 9 zeigt ein einfaches Beispiel eines Gozintographen. Für komplexe Produkte mit mehreren tausend Teilen läßt sich der Graph zwar nicht mehr zeichnen, kann aber für verschiedene Algorithmen benutzt werden. Dabei muß nur jeweils ein Bruchteil des Graphen im Speicher gehalten werden. Der Rest kann jederzeit wieder neu berechnet werden.

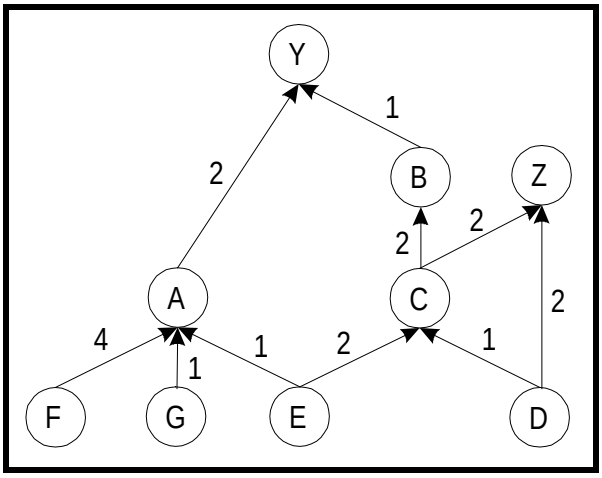


Abbildung 9: Erzeugnisstrukturen als Gozinto-Graph
(aus [Kur93], S. 64, Abb. 2.2-3)

Aus den Erzeugnisstrukturdaten lassen sich durch Traversierung des Graphen ***Stücklisten*** und ***Teileverwendungsnachweise*** berechnen. Stücklisten betrachten ausgehend von einem Teil dessen Kinder im Graphen. Beispielsweise geben Baukastenstücklisten für ein Teil alle direkt in das Teil eingehenden Teile an, Strukturstücklisten berechnen auch alle indirekt eingehenden Teile und Mengenübersichtsstücklisten fassen jeweils gleichartige Teile der Strukturstückliste zusammen, so daß für jedes benötigte Teil dessen Menge abgelesen werden kann. In Teileverwendungsnachweisen (TV) wird der Graph von seinen Blättern aus betrachtet. Analog zu Stücklisten lassen sich Baukasten-TV, Struktur-TV und Mengenübersichts-TV berechnen.

Arbeitspläne beinhalten die Anweisungen, wie ein Teil aus seinen Bestandteilen zusammenzusetzen ist. In einem Arbeitsplan ist festgelegt, in welcher Reihenfolge ***Arbeitsgänge (Operationen)*** durchgeführt werden müssen, auf welchem Arbeitsplatz dies erfolgen muß und wieviel Zeit das Rüsten, Liegen und die Bearbeitung eines Teils in Anspruch nehmen. Weiterhin werden Angaben über den frühesten und spätesten Termin, zu dem die Bearbeitung begonnen werden kann bzw. muß, benötigt. Im Prinzip sind Arbeitspläne sehr ähnlich zu Stücklisten, die die Zusammensetzung eines Teils beinhalten. Ebenso ist die Beschreibung eines Arbeitsganges ähnlich zur Beschreibung eines Teils. Allerdings ist eine Stückliste relativ statisch, da an der Art und Menge der verwendeten Teile selten Änderungen vorgenommen werden können. In einem Arbeitsplan können jedoch Wahlmöglichkeiten hinsichtlich des Arbeitsplatzes und der Reihenfolge vorliegen. Da die Stückliste den Bedarf an Materialressourcen und der Arbeitsplan den Bedarf an übrigen Ressourcen beinhaltet, können beide zu einer ***Ressourcenliste*** zusammengeführt werden.

Arbeitsplatzdaten beschreiben die Betriebsmittel, z.B. einzelne Maschinen. Für diese werden technische Daten, Produktionsdaten usw. festgehalten. Analog zu Arbeitsplänen und Arbeitsgängen sind ***Fertigungsstrukturen*** die Zusammenfassung von Arbeitsplätzen. In Fertigungsstrukturen werden einzelne Arbeitsplätze zu Arbeitsplatzgruppen (Maschinengruppe), Arbeitsplatzgruppen zu Werkstätten o.ä. gruppiert. Ebenso können Arbeitsplätze zu Kostenstellen zusammengefasst werden. In einem Arbeitsplan kann anstelle eines Arbeitsplatzes, auf dem ein Arbeitsgang durchzuführen ist, ebenso eine Arbeitsplatzgruppe notiert sein, was bedeutet, daß bei der Planerstellung ein in dieser Gruppe enthaltener Arbeitsplatz ausgewählt werden muß. Analog zu Arbeitsplätzen werden Daten über das Personal verwaltet. Bei den Personaldaten spielt jedoch noch die Verwaltung der Arbeitszeitdaten (Schichtpläne)

[5] Der Name leitet sich aus dem Englischen „goes into“ ab.

eine große Rolle. In Ressourcenlisten und Fertigungsstrukturen können auch Alternativen eingebaut werden, die Wahlmöglichkeiten bei der Produktion zulassen.

Ein ***Auftrag*** ist die Sepzifikation zur Fertigung eines Zwischen- oder Endprodukts. Für einen Auftrag werden zum einen Daten über Mengen, Termine und Kostenkalkulation benötigt, zum anderen beinhaltet ein Auftrag Verweise auf die Ressourcenlisten der zu fertigenden Teile und somit Verweise auf die benötigten Fertigungsstrukturen, Arbeitspläne und Erzeugnisstrukturen. Die Erstellung eines Ressourcenbelegungsplans ist somit die Festsetzung aller Wahlmöglichkeiten, die in einem Auftrag vorliegen, und die Terminierung der Arbeitsgänge aller zu betrachtenden Aufträge.

Für genauere Erläuterungen zu den Grunddaten von PPS-Systemen, insbesondere zu Datenbankschemata für die vorgestellten Datenstrukturen, wird auf [Kur93] und [Ker94] (jeweils Kapitel 2) verwiesen.

4.2.2. Erstellung von Ressourcenbelegungsplänen in PPS-Systemen

Seit den 60-er Jahren existieren kommerzielle Systeme zur Produktionsplanung. Die ersten Ansätze zur Maschinenbelegungsplanerstellung stammten aus dem Bereich der Operations-Research-Methoden. Diese Methoden können zwar theoretisch eine optimale Belegung finden, in der Praxis ist dieser Ansatz allerdings gescheitert, da wegen der hohen Komplexität des Problems nur bei kleinen Größenordnungen und restriktiven Voraussetzungen Optimierungsverfahren möglich sind ([Kur93], S.41). Die folgende Zusammenstellung über verschiedene Ansätze zur Planerstellung ist aus [Hie96] entnommen.

MRP

Die in vielen heutigen PPS-Systemen verwendeten Planungsalgorithmen beruhen auf der MRP (Materials Requirement Planning) - Philosophie. Der älteste und einfachste Ansatz von MRP erlaubt die computergestützte Auflösung der Stücklisten zur Materialbedarfsermittlung. Ausgehend von Stücklisten, Teileverwendungslisten, dem vorhandenen Bestand und dem „Master Production Schedule“ (MPS) - einem Grobplan - werden die Brutto- und Nettobedarfe ermittelt. Daraus können anschließend Bestellmengen, Losgrößen und Vorlaufverschiebung berechnet werden. Das MRP-Verfahren in seiner einfachsten Form geht von unendlichen Kapazitäten aus. Da nicht berücksichtigt wird, ob es zu Überlastungen kommt, wird zu den eigentlichen Bearbeitungszeiten der Operationen ein Puffer addiert, der eventuell entstehende Engpässe vermeiden soll.

MRP-II

Da die Nichtberücksichtigung der vorhandenen Kapazitäten in MRP einen großen Nachteil darstellt, wurde MRP zum Nachfolgekonzept MRP-II (Manufacturing Resources Planning) erweitert. MRP-II umfaßt ein Modul für die Materialbedarfsplanung, das dem ursprünglichen MRP entspricht, und einem zweiten Modul zur Maschinenbelegungsplanung, das nicht mehr von unendlichen Kapazitäten ausgeht. Der jeweilige Kapazitätsbedarf und das vorhandene Kapazitätsangebot werden getrennt voneinander ermittelt und gegenübergestellt, wodurch Überlastungen und Freiräume deutlich werden. Eine Konfliktlösung durch einen Kapazitätsausgleich kann daraufhin in einem weiteren Modul durchgeführt werden, doch gehört ein Kapazitätsausgleichsmodul nicht zum ursprünglichen Umfang von MRP-II. In den meisten heutigen PPS-Systemen, die auf MRP-II beruhen, wird die Maschinenbelegungsplanung durch die Strategien der Vorwärts- und Rückwärtsterminierung durchgeführt:

- **Rückwärtsterminierung:** Ausgehend vom spätesten Termin des Auftrags wird die letzte auszuführende Operation betrachtet. Der späteste Termin dieser Operation wird berechnet, indem vom spätesten Endtermin des Auftrags die Dauer der Operation (also Transport,- Liege-, Bearbeitungs- und Rüstzeit) abgezogen wird. Als Starttermin dieser Operation wird ein Termin gesucht, der so knapp wie möglich vor diesem ST liegt und alle anderen Nebenbedingungen

erfüllt. Dieses Vorgehen wird für alle vorangehenden Operationen sukzessive durchgeführt, indem der ST und der Starttermin einer Operation aus dem Starttermin der nachfolgenden Operationen berechnet wird. Sind alle Operationen eingeplant, so erhält man einen Starttermin für den Auftrag und die Bedarfstermine für die benötigten Materialien. Die Rückwärtsterminierung hat den Vorteil, daß sie für den jeweiligen Materialbedarf einen spätesten Termin errechnet, was in Bezug auf die Lagerhaltung wirtschaftlich ist. Allerdings besteht die Gefahr des Terminverzugs, da für den Materialbedarf ein Termin bestimmt wird, der davon ausgeht, daß die benötigten Materialien kurzfristig und wirtschaftlich beschafft werden können.

- **Vorwärtsterminierung:** Die Vorwärtsterminierung verläuft analog zur Rückwärtsterminierung, nur daß dabei von einem frühesten Termin für den Auftrag nach vorne gerechnet wird. Die Vorwärtsterminierung läßt für spätere Umplanungen möglichst viel Spielraum, ist aber unwirtschaftlicher, da möglicherweise Material und Teile unnötig lange und früh benötigt werden.

Graphisch lassen sich die erstellten Belegungspläne durch Plantafeln, die als Gantt-Diagramme Termine und Dauern der einzelnen Operationen auf einer Ressource zeigen, und als Auslastungsgebirge über Bedarf und Angebot für eine Ressource über einen Zeitraum darstellen.

Scheduling durch Simulation

Einen anderen Ansatz stellt das Scheduling durch Simulation dar. Dabei wird ein Simulationsmodell der Fertigungsumgebung aufgebaut, das die technologischen Notwendigkeiten der Fertigung widerspiegelt. Im Simulationsmodell werden den einzelnen Arbeitsplätzen Warteschlangen für die zu bearbeitenden Operationen zugewiesen und mit einer bestimmten Prioritätsregel (Warteschlangendisziplin) versehen. Daraufhin wird eine Simulation des Fertigungsprozesses durchgeführt.Ausgehend von einem Startzeitpunkt werden alle Operationen in die Warteschlangen der jeweiligen Arbeitsplätze eingereiht. Welche Operation als erste bearbeitet wird, entscheidet die Warteschlangendisplin. Diese Operation wird aus der Warteschlange entfernt und belegt für die Dauer der Operation den Arbeitsplatz. Nach der Bearbeitung der Operation wird der Arbeitsplatz freigegeben und die nächste Operation aus der Warteschlange entnommen. Die sich daraus ergebende Abfolge der Operationen auf den Ressourcen ergibt einen Ressourcenbelegungsplan. In [Zel92] wird dieser Einsatzbereich für Simulationsmodelle als ***Online-Unterstützung der dispositiven Steuerung*** bezeichnet.

Scheduling durch Simulation ist somit eine konstruktive Lösungsmethode, da der Maschinenbelegungsplan schrittweise während der Simulation aufgebaut wird. Die gewählten Warteschlangendisziplinen sind Heuristiken, die eine Auswahl der Reihenfolge auf der lokalen Ebene der einzelnen Maschinen bzw. der Maschinengruppe erlauben, wobei der Schedule in zeitlich fortschreitender Weise konstruiert wird. [Mos93] gibt die in Tabelle 3 aufgestellte Klassifikation für Prioriätsregeln an. Weiterhin sind zusammengesetzte Prioritätsregeln möglich, die sich aus einer Gewichtung einzelner Regeln ergeben. [Mos93] bietet eine vergleichende Bewertung von verschiedenen Prioritätsregeln in einem Simulationsmodell, in dem anhand statistischer Kriterien Aufträge erzeugt werden. Die Bewertung der Prioritätsregeln erfolgt hauptsächlich unter dem Gesichtspunkt der Termintreue.

In Abschnitt 7.2 wird gezeigt, wie in DYBBS Scheduling durch Simulation durchgeführt werden kann und wie heuristische Prioritätsregeln dort in ein Simulationsmodell eingebaut werden können. Im Gegensatz zu dem in DYBBS verfolgten Ziel der Bewertung eines Belegungsplanes durch eine Simulation der Fertigungsumgebung, in der die Planvorgaben durch Störungen u.a. nicht eingehalten werden können, spielen für das Scheduling durch Simulation stochastische Einflüsse während der Fertigung eine untergeordnete Rolle. In den meisten Fällen erfolgt das Erstellen des Belegungsplans unter rein deterministischen Gesichtspunkten. Es ist allerdings auch möglich, stochastische Einflüsse zu berücksichtigen, indem z.B. die Auftragserzeugung zufallsbasiert erfolgt.

Wirkungsbereich	Orientierung	Beispiel
Lokal	terminorientiert	Kürzeste Pufferzeit SL (slack)
	bearbeitungszeitorientiert	Kürzeste Restbearbeitungszeit LWKR (least work remaining)
	wartezeitorientiert	Längste Wartezeit FIFO (first in first out)
Global	zugangsorientiert	Frühester Endzeitpunkt FOFO (first off first on)
	durchlaufzeitorientiert	Folge-Warteschlange mit geringstem Arbeitsvorrat WINQ (work in next queue)

Tabelle 3: Klassifikation heuristischer Prioritätsregeln (nach [Mos93], Kap. 3.1)

OPT

Das OPT (Optimized Production Technology) - Konzept geht im Gegensatz zu den MRP-Verfahren von einer simultanen Planung aller Faktoren aus, wobei das Ziel verfolgt wird, die Aufträge engpassorientiert einzuplanen. Zur Planung werden aus den Maschinen, Produkten und Materialien Belastungsprofile für die einzelnen Maschinen erstellt, aus denen die aktuellen Engpässe ersichtlich werden. Alle Engpässe und die darauf folgenden Arbeitsgänge werden als „kritische Bereiche“ durch Vorwärtsterminierung bestimmt. Alle vor den Engpässen liegenden Arbeitsgänge werden anschließend als „nicht-kritische Bereiche“ durch Rückwärtsterminierung eingeplant. Dadurch entsteht eine Planung um die Engpässe herum. Die Probleme mit OPT ergeben sich daraus, daß die erstellte Zuordnung sehr exakt eingehalten werden muß und daß Engpässe sich im Verlauf der Planung verschieben können..

4.2.3. Wissensbasierte Erstellung von Ressourcenbelegungsplänen

In diesem Abschnitt wird die Erstellung von Ressourcenbelegungsplänen mit wissensbasierten Ansätzen betrachtet, die in konventionellen PPS-Systemen i.a. noch nicht vorhanden sind.

Die Erstellung eines Ressourcenbelegungsplans kann als Spezialfall des Problemtyps ***Zuordnung*** angesehen werden ([Pup90], S. 128ff). In einer Zuordnung wird eine Menge von Angebotsobjekten auf eine andere Menge von Nachfrageobjekten abgebildet, wobei Randbedingungen eingehalten werden müssen. Müssen den Nachfrageobjekten auch Zeitintervalle zugeordnet werden, d.h. die Zeit tritt als Angebotsobjekt mit einer bestimmten Kapazität auf, so spricht man von ***Scheduling*** oder ***Ablaufplanung***. Sind beide Mengen gleich groß mit Umfang *n*, so existieren *n!* verschiedene Zuordnungen. Da für einen Ressourcenbelegungsplan oft mehrere tausend Arbeitsgänge eingeplant werden müssen, scheidet ein systematisches Ausprobieren aller Lösungen zur Bestimmung einer optimalen Zuordnung aus.

Die Strategie „Vorschlagen-und-Vertauschen“

Als Problemlösungsmethode für den Problemtyp Zuordnung schlägt [Pup90] die Strategie „Vorschlagen-und-Vertauschen“ vor, die in Abbildung 10 dargestellt ist. Der grundsätzliche Ablauf ist dabei der folgende ([Hes95], S. 6):

1. Zuerst wird das nächste einzuplanende Nachfrageobjekt ausgewählt. Dabei können unterschiedliche Planungsstrategien durch Angabe der Auswahlregeln implementiert werden.
2. Für dieses Nachfrageobjekt wird ein passendes Angebotsobjekt bestimmt.
3. Nach jedem solchen Zuordnungsschritt wird überprüft, ob hinreichend wichtige Restriktionen verletzt sind. Diese Verletzungen können durch Vertauschungs- und Korrekturregeln behoben werden.
4. Der Algorithmus terminiert durch eine vollständige Zuordnung aller Objekte. Falls für den Zuordnungslauf ein Zeitlimit eingeführt wurde, entsteht ein suboptimaler Plan.

Einer der Vorteile des Algorithmus besteht darin, daß die Anzahl der getroffenen Zuordnungen monoton wächst, da durch das Vertauschen nie ein Zuordnungsschritt einfach gelöscht wird, ohne daß ein neuer dazukommt. Für die Anwendung des Algorithmus wird Wissen über eine geeignete Auswahl von Nachfrager und Anbieter sowie verschiedene Korrekturregeln benötigt. Die Zuordnungsshell COKE (s.

Abschnitt 6.1) verwendet u.a. diese Strategie zur Lösung verschiedener Zuordnungsprobleme, z.B. für Schulstundenpläne (REST) und Ressourcenbelegungspläne (WIZARD). Weitere Erläuterungen zum Problemtyp Zuordnung und zum Algorithmus „Vorschlagen-und-Vertauschen" befinden sich in [Pup90] und [Poe95].

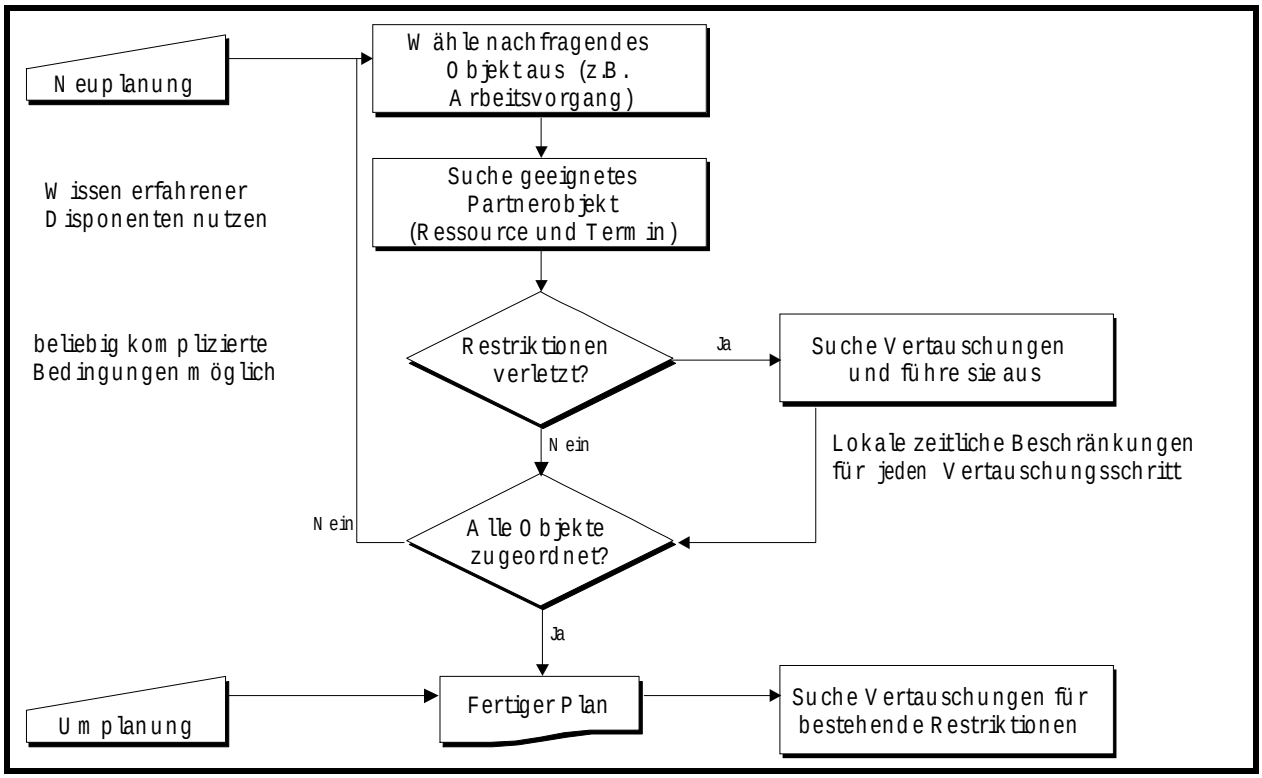


Abbildung 10: Vorschlagen-und-Vertauschen-Strategie (aus [Hes95], S. 5)

Für die Ressourcenbelegungsplanung sind die Ressourcen Arbeitsplatz, Personal, Material, Werkzeug und Zeit die Anbieter, die eine bestimmte Kapazität oder Menge bereitstellen, während Aufträge und Arbeitsgänge als Nachfrager bestimmte Kapazitäten beanspruchen. Als Rahmenbedingungen treten dabei u.a. zugesagte Endtermine für Aufträge, Anliefertermine für Material, Kapazitäten und Eigenschaften für Maschinen und Reihenfolgen von Arbeitsgängen auf. Eine besondere Angebotsressource ist die Zeit, da jeder Nachfrager eine Ressource für eine bestimmte Dauer und zu einem bestimmten Termin benötigt. Diese besondere Stellung läßt sich daran ablesen, daß i.a. in PPS-Systemen und Lehrbüchern zwischen Materialwirtschaft, d.h. der Planung der Bereitstellung von Material in der richtigen Menge zum richtigen Termin, auf der einen Seite und Zeitwirtschaft, also der Planung von Terminen und Kapazitäten, auf der anderen Seite unterschieden wird ([Ker94], [Kur93]). Trotzdem bleibt die Forderung nach einer simultanen Planung beider Aspekte bestehen.

[Hes97-2] erläutert die Anwendung der „Vorschlagen-und-Vertauschen"-Strategie auf die Ressourcenbelegungsplanung. Dabei ergibt sich das Problem, daß es sich bei der Ressourcenbelegungsplanung um ein kontinuierliches Zuordnungsproblem handelt. Es muß für die Zeitachse eine geeignete Intervallarithmetik über belegte und freie Zeitintervalle der einzelnen Ressourcen gefunden werden, um die „Vorschlagen-und-Vertauschen"-Strategie anzuwenden. Der Auswahlschritt erfolgt in WIZARD so, daß der Nachfrager terminorientiert, der Anbieter hauptsächlich auslastungsorientiert gesucht wird. Zur Korrektur verletzter Restriktionen wird ein mehrstufiges Verfahren verwendet, das die an den Verletzungen beteiligten Nachfrager ermittelt, diese auf ihre Verschiebbarkeit hin bewertet und Verschiebungen durchführt. Dieses Verfahren kann sowohl zur Erstellung eines Planes als auch zur Umplanung oder Optimierung eines bestehenden Planes verwendet werden.

Weitere wissensbasierte Ansätze

Seit 1980 wurden verschiedene Scheduling-Systeme entwickelt, die eine Constraint-basierte Konstruktion der Zuordnung durchführen. Diese Systeme können simultan alle Faktoren bei der Planerstellung berücksichtigen. Die konstruktive Planerstellung führt dabei eine Suche durch den Raum aller möglichen Zuordnungen durch, wobei die bereits getroffenen Schritte und die Constraints den nächsten Zuordnungsschritt bestimmen. Als Suchstrategien werden Hill-Climbing oder Best-First mit Backtracking verwendet. Die Constraints ergeben sich z.B. aus technologischen Notwendigkeiten, Kapazitätsgrenzen und dem Prinzip der Gewinnmaximierung. Verschiedene Schedulingsysteme mit konstruktiven Lösungsmethoden (ISIS, OPIS, Micro-Boss) werden in [Hie96] vorgestellt.

Muß jedoch ein bereits vorhandener Schedule umgebaut werden, so bietet sich zu diesem Zweck eine weitere Klasse von Lösungsansätzen an, die auf der „Reparatur“ eines Planes durch kleine Veränderungen beruhen. In diesen Verfahren können auch zugeordnete Objekte, die keine Constraintverletzungen aufweisen, umgelegt werden, um für andere Objekte, deren Constraints verletzt sind, Platz zu machen. Die reparaturbasierten Verfahren arbeiten nur auf dem Suchraum der vollständigen Pläne, wodurch sie globale Constraints besser überprüfen können als konstruktive Methoden. Daher eignen sich die reparaturbasierten Verfahren auch zur Optimierung eines zuvor durch eine konstruktive Methode erstellten Planes. Einzelne reparaturbasierte Planungssysteme werden in [Hie96] beschrieben.

4.3. Einsatzgebiete für Simulation in PPS-Systemen

Neben der im vorigen Abschnitt erläuterten Möglichkeit, durch Simulation eines Fertigungssystems Belegungspläne zu erstellen, bieten sich im Fertigungsumfeld weitere Einsatzgebiete für Simulationsanwendungen. [Sch87] nennt als Anwendungsgebiete für Simulation in der Werkstattsteuerung folgende Bereiche:

- **Verifikation der Prozeßablaufplanung:** Durch Simulation des Produktionsprozesses kann die Einhaltung vorgebener Randbedingungen wie begrenzte Pufferkapazitäten oder Mindestbestände überprüft werden.
- **Bewertung der geplanten Prozeßführung:** Liegen mehrere Planvarianten vor, so kann die geeigneste Alternative ermittelt werden. Zur Bewertung können dabei sowohl statische Grundziele wie die Einhaltung der Endtermine als auch dynamische Eigenschaften wie eine gleichmäßige Auslastung verwendet werden.
- **Optimierung der Prozeßablaufplanung:** Ausgehend von der Bewertung der Prozeßführung können verschiedene Planvarianten erzeugt werden, um die beste Alternative zu ermitteln.
- **Unterstützung der Ressourceneinsatzplanung:** Durch Simulation können für die Planung Erkenntnisse über die auftretenden Ort- und Zeitbedarfe von Personal, Rüst- und Transportvorgängen gewonnen werden.
- **Test von Strategien zur Störfallbeseitigung:** Vorausschauend können Alternativen geprüft werden, wie Störfälle im Produktionsprozeß behandelt werden müssen. Insbesondere können Erkenntnisse darüber gewonnen werden, unter welchen Umständen eine Umplanung notwendig wird.
- **Schulung von Betriebspersonal:** Durch das Durchspielen von Situationen können die Disponenten auf eine sichere Beherrschung ihrer Aufgaben trainiert werden, was u.a. im Hinblick auf die Handhabung von Störfällen erfolgen kann.

Als übergeordneter Punkt zur Verifikation wäre zusätzlich die Fertigungsplanung zu nennen, in der der Entwurf von Fabrikhallen und Fertigungszellen im Vordergrund steht. Dieser Gesichtspunkt steht im besonderen bei einer Fließproduktion mit einem hohen Automatisierungsgrad im Vordergrund. In diesen Fertigungssystemen besteht ein hoher Bedarf nach einer optimalen Anordnung der einzelnen Arbeitsplätze und deren Verbindung durch Materialtransportsysteme. Da dieser Aufgabenbereich durch kommerzielle Simulationssysteme (s. Abschnitte 5.2 und 5.3) mit ihren vielfältigen Modellbibliotheken sehr gut abgedeckt ist, gehört dieses Anwendungsgebiet nicht zu den Zielen des im Rahmen dieser Arbeit

implementierten Simulationssystems DYBBS. Einsatzgebiete für DYBBS ergeben sich in den anderen fünf aufgezählten Punkten, wobei insbesondere die Bewertung und die Störfallbeseitigung im Vordergrund stehen.

4.4. Statische Bewertungskriterien für Ressourcenbelegungspläne

Die Bewertung eines Belegungsplanes erfolgt unter den Zielen, die für ein PPS-System angestrebt werden. Nach [Ker94] sind die vier Grundziele eines PPS-Systems:

- hohe Kapazitätsauslastung
- kurze Durchlaufzeiten
- geringe Lagerbestände
- hohe Lieferbereitschaft

Diese vier Ziele sind möglichst gleichzeitig zu erreichen, was schwierig ist, da die Optimierung auf ein Ziel hin den anderen Zielen entgegenläuft. In diesem Abschnitt werden Bewertungsansätze vorgestellt, die einen vollständigen Belegungsplan bewerten. In diesen traditionellen Ansätzen wird jedoch nicht die zeitliche Dynamik, die sich bei der Durchführung des Belegungsplanes ergibt, berücksichtigt. Daher werden diese Bewertungsansätze statisch genannt. In Abschnitt 4.6 werden Ansätze für dynamische Bewertungskriterien vorgestellt.

Zur Bewertung eines Planes können zwei grundsätzliche Kategorien herangezogen werden. Die ***Korrektheit*** bewertet, wie genau die grundlegenden Anforderungen an einen Ressourcenbelegungsplan erfüllt sind, d.h. diese Kriterien sind harte Einschränkungen. Zu ihnen gehören:

- **Vollständigkeit:** Alle Operationen müssen eingeplant werden. Diese Forderung bedeutet u.a., daß alle Daten, die für die Planerstellung benötigt werden, komplett vorhanden sein müssen. Dies kann problematisch sein, wenn das zu fertigende Produkt eine Neukonstruktion darstellt, so daß über einige Arbeitsgänge noch keine Daten vorliegen. Genauso wichtig ist die Erhaltung der Konsistenz in den Teilestamm- und Arbeitsgangdaten, die allein aufgrund ihrer Größe aufwendig und schwierig ist.
- **Einhaltung der Reihenfolge:** Die Reihenfolge, in der verschiedene Arbeitsgänge eines Auftrags ausgeführt werden können, ist oft technologisch vorgegeben. Eine Änderung der Reihenfolge darf somit nur erfolgen, wenn sie explizit in den Daten erlaubt wird. Ebenso kann die Dauer von Rüst- und Bearbeitungszeit aus technologischen Gründen nicht verändert werden.
- **Einhaltung der existierenden Terminbedingungen:** Der vom Kunden vorgegebene Endtermin des Auftrags und die sich daraus ergebende ST der Operationen sollten nicht überschritten werden. Liegen für Materialanlieferungen Termine vor, so können die FT der dazugehörenden Operationen in der Regel nicht unterschritten werden.
- **Vermeidung von Kollisionen:** Überschneiden sich auf einer Ressource Operationen, so bedeutet dies, daß zwei Arbeitsgänge zumindest in Teilen gleichzeitig ausgeführt werden sollen. Für Ressourcen, die nicht speziell für solche Fälle geschaffen sind (z.B. sog. Mehrmaschinenbediener, d.h. Werker, die gleichzeitig mehrere Maschinen, die für eine gewisse Dauer ohne Beaufsichtigung arbeiten können, bedienen können), stellt dies eine Unmöglichkeit dar. In der PPS-Praxis sind jedoch oftmals Überschneidungen bis zu einem gewissen Prozentsatz erlaubt. Das Erlauben von Überschneidungen ist das durch die Erfahrung bedingte Eingeständnis, daß die Angaben des PPS-Systems nur Schätzungen sind, die noch mit einem Sicherheitsspielraum versehen sind. Deshalb kann davon ausgegangen werden, daß die Überschneidungen sich in der Realität von selbst auflösen und Operationen „hineingequetscht“ werden können.
- Als eine weitere harte Einschränkung kann die **Einhaltung von vorbeugenden Wartungsarbeiten** betrachtet werden, die in der Regel in gleichmäßigen Abständen durchgeführt werden müssen. Dabei ist jedoch der Termin für diese Arbeiten nicht fest, sondern kann in einem ge-

wissen Rahmen verschoben werden. Somit kann dieses Kriterium auch zu den weichen Constraints gezählt werden.

In der Praxis ist aufgrund der großen Anzahl der einzuplanenden Operationen kaum ein Plan in diesem Sinne korrekt. Eine reine Spezialisierung auf korrekte Pläne und einen daraus sich ergebenden unproblematischen Ablauf der Produktion würde dazu führen, daß FT und ST sehr weit auseinandergezogen werden, was zu einer unwirtschaftlichen Bevorzugung des Grundziels Termintreue führt.

Die zweite Kategorie zur Bewertung von Ressourcenbelegungsplänen ist die ***Optimalität*** des Planes. Die Optimierung eines Planes erfolgt nach den vier oben genannten Grundzielen, wobei die Korrektheit des Planes nicht verschlechtert werden sollte. Die Optimalitätskriterien sind somit weiche Einschränkungen. Es bieten sich folgende Gesichtspunkte an:

- **Minimierung der Rüstzeiten:** Je nachdem, in welcher Reihenfolge die Operationen auf einer Ressource abgearbeitet werden, fallen unterschiedliche Rüstzeiten an. Wie und in welchem Umfang umgerüstet werden muß, ist dabei eine durch die Technik vorgegebene Tatsache. Eine Minimierung der Rüstzeiten verringert die Durchlaufzeit (DLZ) der Aufträge, die durch die Dauer zwischen dem Anfang der ersten Operation und dem Ende der letzten Operation definiert ist. In die DLZ gehen sowohl die Bearbeitungszeiten als auch die Wartezeiten ein.
- **Verringerung der Wartezeiten:** Wartezeiten ergeben sich aus der Pause zwischen der Beendigung einer Operation und dem Beginn der nächsten. Geringe Wartezeiten verringern die Durchlaufzeit und die Lagerkosten. Untersuchungen ergaben, daß die Wartezeiten bis zu 80% der DLZ ausmachen ([Ker94], S. 169), so daß an dieser Stelle ein großes Optimierungspotential besteht. Die Wartezeiten können aus Auftrags- oder Ressourcensicht betrachtet werden, so daß sich zwei Ansätze zur Minimierung der Wartezeiten ergeben, bei dem der eine Ansatz einen größeren Wert auf die DLZ, der andere auf die Auslastung legt.
- **„Alles so spät wie möglich“:** Die Rückwärtsterminierung versucht jede Operation zum spätest möglichen Zeitpunkt einzuplanen. Ein möglichst später Zeitpunkt ermöglicht die Berücksichtigung von Änderungen, die sich bis zum Start der Operation ergeben, und senkt Lagerkosten und DLZ. Allerdings kann dieses Prinzip sich nachteilig auf die Termintreue auswirken.
- **Erhöhung der Kapazitätsauslastung:** Da Maschinen in ihrer Anschaffung teure Ressourcen sind, sollten sie so häufig wie möglich genutzt werden. Aus der Sicht einer solchen Maschine soll der Plan daher möglichst ohne Pausen sein. Dabei kann eine Beschränkung auf teure (wichtige, neue) Maschinen bei der Kapazitätsoptimierung sinnvoll sein.
- **Billige Arbeitszeiten:** Für die Ressource Personal sind verschiedene Zeiten unterschiedlich teuer. Für Überstunden, Nacht- und Sonderschichten müssen Zulagen bezahlt werden und Verhandlungen mit dem Betriebsrat geführt werden. Daher ist bei der Planerstellung die Inanspruchnahme dieser Sonderzeiten zu minimieren.
- **Minimierung der momentanen Arbeit:** Hierbei soll die mittlere Anzahl von Aufträgen, die zu einem Zeitpunkt bearbeitet werden, möglichst gering sein. Eine geringe Anzahl gleichzeitig aktiver Aufträge verringert die Zwischenlagerhaltung.

Die vorgestellten Kriterien stellen nur eine Auswahl der Möglichkeiten dar. Dabei sind bestimmte Kriterien auch auf bestimmte Fertigungstypen anwendbar. Für eine weitgehend automatisierte Fertigung ist die Auslastung der Maschinen wichtiger als die Arbeitszeiten der (wenigen) Werker. Eine Produktion auf Lager hat wenig, ein Auftragsfertiger sehr viel Interesse an einer hohen Termintreue.

Für die Berechnung der Bewertungskriterien werden folgende Daten über jede Operation benötigt:

- *Rüstzeit* t_r
- *Stückzeit* t_e
- *Menge* m

- *Liegezeit t_l*
- *Transportzeit t_t*
- *frühester Termin FT*
- *spätester Termin ST*
- *Plantermin RT*

Es ergibt sich die Bearbeitungsdauer durch $T_b = t_r + m \cdot t_e + t_l + t_t$. Die Stück- und Liegezeit sowie die Menge ist fest vorgegeben. Die Rüst- und Transportzeit ergeben sich allerdings erst bei der Einplanung. Die Rüstzeit ist abhängig von der zuvor auf dieser Ressource bearbeiteten Operation. Die Transportzeit hängt von der Ressource ab, auf der die Nachfolgeroperation bearbeitet wird. Für beide ist die Datenangabe durch eine Matrix möglich, allerdings bedingt deren Erstellung und Pflege einen erheblichen Aufwand. Transportzeitmatrizen werden relativ häufig verwendet, da zwischen den einzelnen Arbeistplätzen in einer Fabrikhalle oft nur relativ wenig sinnvolle Wege angegeben werden können. Für Rüstzeitmatrizen wird oft auf der stark vergröberten Ebene der Materialklassen gearbeitet. Der Zeitraum zwischen der Beendigung der Vorgängeroperation und dem Beginn der Rüstzeit wird als Wartezeit T_w der Operation bezeichnet. Die Durchlaufzeit (DLZ) der Operation ergibt sich damit als $DLZ = T_b + T_w$. Durch Aufsummierung über alle Operationen eines Auftrags lassen sich die Bearbeitungszeit und DLZ einer Operation berechnen.

Die Auslastung einer Ressource über einen Zeitraum kann durch den Anteil der Bearbeitungszeiten aller in diesem Zeitraum bearbeiteten Operationen berechnet werden. Die Auslastung eines Auftrags ergibt sich aus der Bearbeitungszeit des Auftrags geteilt durch dessen DLZ. Für die Überprüfung der Termintreue werden die vereinbarten spätesten Endtermine SET ($SET=ST+T_b$) der Aufträge mit den tatsächlichen Endterminen verglichen.

Für die Minimierung der Lagerbestände sind die Lagerhaltungskosten und Losgrößen zu betrachten. Diese Optimierung spielt allerdings bei der Mittelfristdisposition eine wichtigere Rolle als bei der Kurzfristdisposition, bei der in der Regel die Materialliefertermine bereits bekannt sind. Daher lassen sich kurzfristige Belegungspläne nicht besonders auf dieses Grundziel hin optimieren. Die Berechnung von Lagerkosten und Formeln für Losgrößen werden in [Ker94] (Kapitel 4) beschrieben.

Ähnlich zu den vorgestellten Maßen für DLZ, Auslastung, Termintreue und Lagerbestand können weitere Bewertungskriterien definiert werden. Diese werden als ***Kennzahlen*** bezeichnet. [Hac89] stellt im Zusammenhang mit dem Analysewerkzeug FANAL weitere Kennzahlen vor, die in diesem Werkzeug definiert und analysiert werden können. Diese Kennzahlen umfassen u.a. den Auftragsrückstand, den Wiederholanteil, Wartezeiten, Losgrößen und Reihenfolgeänderungen. Diese Kennzahlen dienen in FANAL dem Zweck, um Abweichungen zwischen Plan und Ablauf in systematische und unsystematische Abweichungen zu gliedern, wobei Wert auf die systematischen Abweichungen gelegt wird. Diese deuten auf Fehler in den Daten oder der Planerstellung hin. Unsystematische Abweichungen werden durch Störungen und Zufälle hervorgerufen und *„... sind nicht dem Fertigungssteuerungssystem anzulasten“* ([Hac89], S. 292). Eine Berücksichtigung dieser Störeinflusse bei der Planerstellung wird in FANAL nicht angestrebt, im Gegensatz zur Zielsetzung in DYBBS.

Die vorgestellten Maße für DLZ, Auslastung, Termintreue und Lagerbestand können auch graphisch veranschaulicht werden. An Durchlaufdiagrammen kann die Termintreue und DLZ abgelesen werden. Dazu trägt man zur Darstellung der Termintreue die Kundenwunsch- und die Ablieferungstermine von Aufträgen ein. Zur Veranschaulichung der DLZ von Arbeitsgängen auf einem Arbeitsplatz trägt man die Anlieferungs- und Abtransporttermine in zwei Kurven ein. Die DLZ einer Operation ergibt sich aus dem horizontalen Abstand zwischen den beiden Kurven. Analog kann man die DLZ von Aufträgen in einer Abteilung oder im ganzen Betriebes darstellen. Der Lagerbestand kann durch die Kurven für Materialzugänge und -abgänge verdeutlicht werden, wobei der vertikale Abstand beider Kurven den Lagerbestand zu einem Zeitpunkt darstellt. Werden Kapazitätsangebot und -bedarf gegeneinander aufgetragen, so ist die nicht nutzbare Kapazität bzw. Überlastung durch den vertikalen Ab-

stand beider Kurven ablesbar. Abbildung 11 zeigt Beispiele für die vorgestellten Diagramme. Kann in der Y-Achse nicht nur der Arbeitsinhalt in Stunden oder Stück angegeben werden, sondern durch eine Kostenfunktion in Geld umgerechnet werden, so ist die Fläche zwischen beiden Kurven ein aussagekräftiges Maß für die dargestellten Ziele. Die Erstellung der Diagramme kann automatisch erfolgen, da dafür nur eine Auswertung der PPS-Datenbanken erfolgen muß.

Aus den genannten Korrektheits- und Optimalitätskriterien ein Gesamtkriterium zu bilden, ist allerdings keine allgemein lösbare Aufgabe. Vielmehr muß ein anpassbares Verrechnungsschema gefunden werden, das die Einzelkriterien zu einer Gesamtbewertung zusammenfaßt. Ein Ansatz, der z.B. in WIZARD (s. Abschnitt 6.1) verwendet wird, ist derjenige, die Bewertungskriterien durch Constraints zu formulieren, für deren Verletzung eine bestimmte Anzahl von Strafpunkten vergeben wird. Die Bewertung eines Planes ergibt sich dann aus der Summe der Strafpunkte.

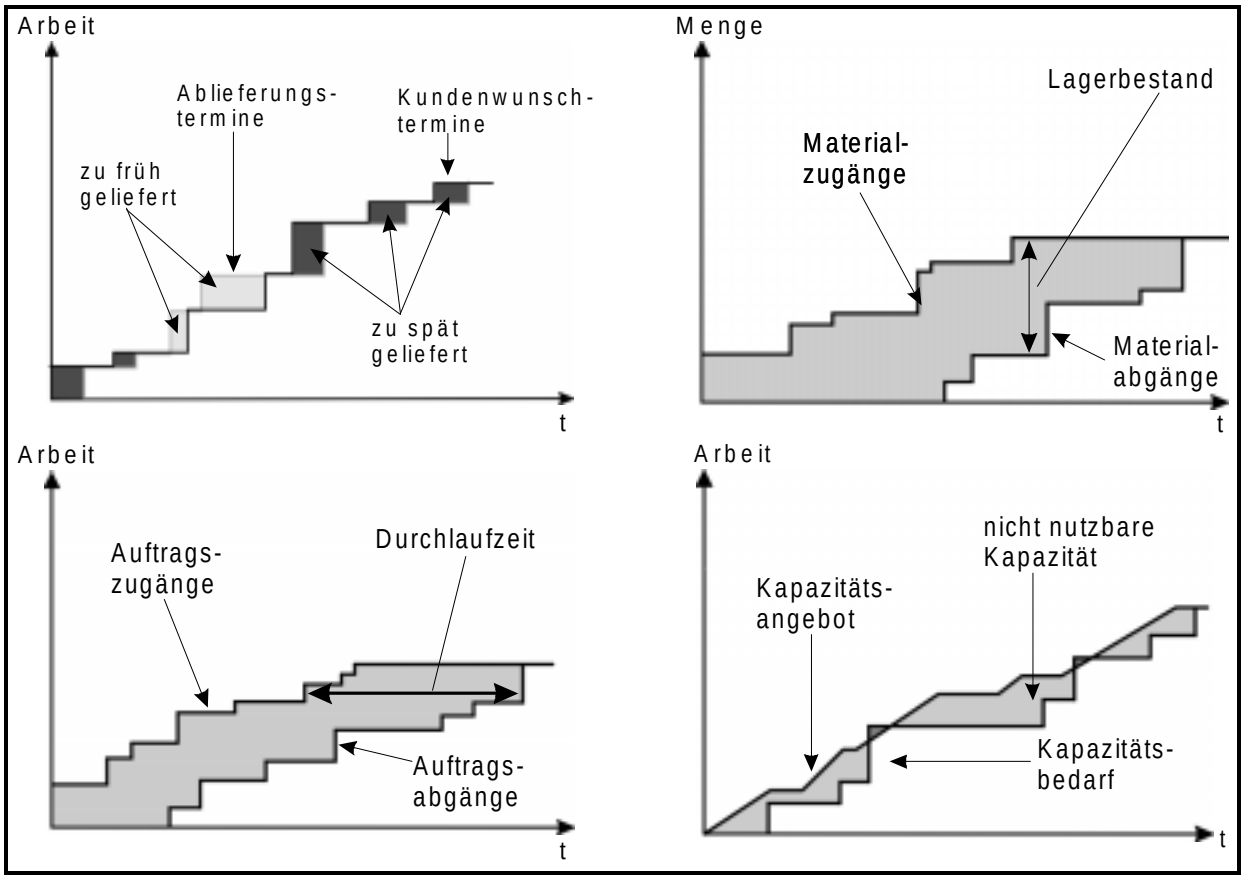


Abbildung 11: Diagramme zur Darstellung von Termintreue, DLZ, Lagerbestand und Auslastung (nach [Ker94], S. 226, Abb. 7.16-7.19)

Liegen mehrere, unterschiedliche Belegungspläne über die gleichen Anbieter und Nachfrager und den gleichen Terminrahmen vor, so kann eine Bewertung aufgrund der oben vorgestellten Bewertungskriterien erfolgen, indem die jeweiligen Unterschiede zwischen den Einzelbewertungen betrachtet werden. Dieser Fall tritt z.B. beim Vergleich zwischen Belegungsplan und realem Produktionsablauf oder zwischen Belegungsplan und Simulationsablauf auf. Beispielsweise wird für das Bewertungskriterium „Wartezeit der Operationen“ für jede Operation für beide Belegungspläne deren Wartezeit berechnet, die Differenz beider Wartezeiten gebildet und diese Differenz statistisch ausgewertet. Während für die einzelnen Belegungspläne hinsichtlich dieses Kriteriums Aussagen der Art „die durchschnittliche Wartezeit auf Ressource X war in Plan A nur halb so groß wie auf Y“ gemacht werden können, erlaubt der Vergleich beider Belegungspläne Aussagen der Art „Plan A hat im Durchschnitt geringere Wartezeiten als Plan B“.

In DYBBS sind zum Vergleich von Simulationsablauf und Belegungsplan anhand statischer Kriterien folgende Bewertungen implementiert worden, die zusätzlich nach einzelnen Ressourcen oder Aufträgen aufgeteilt werden können:

- Bearbeitungszeit, Wartezeit, Durchlaufzeit
- Auslastung
- Pufferzeit
- Starttermine, Endtermine

Einen weitgehenderen Ansatz zum Vergleich mehrerer Belegungspläne wird in [Sch98] vorgestellt. In einem Projekt der TU Illmenau wird ein System zur Bewertung von PPS-Entscheidungen entwickelt, in dem ausgehend von einer Menge von Handlungsalternativen in Form von verschiedenen Belegungsplänen und der Unternehmenssituation für jede Alternative eine Konsequenz prognostiziert wird und diese Prognosen in einer Bewertungskomponente miteinander verglichen werden können. Für Prognose und Bewertung können in diesem System verschiedene Ansätze, z.B. Fuzzy-Logic oder Simulation, verwendet werden.

4.5. Stochastische Eigenschaften des Fertigungsprozesses

In einem Betrieb unterliegen viele Einzelheiten zufälligen Schwankungen. Eine Nichtbeachtung dieser Schwankungen führt zu Fehlmengen für benötigte Materialien, Kapazitätsengpässen und -überschüssen, die zu einer unrentablen Produktion führen. Werden die Schwankungen jedoch durch überhöhte Sicherheitsbestände und -zeiten aufgefangen, so ist dies ebenfalls unwirtschaftlich, da zu hohe Lagerkosten und Kapazitätsüberschüsse entstehen. Daher ist eine genaue Abschätzung der Zufälligkeiten und eine daraus abgeleitete Planung für Kapazitäts- und Sicherheitsbestände unverzichtbar für ein Unternehmen. In diesem Abschnitt soll auf die zufälligen Ereignisse, die für den Bereich der Kurzfristdisposition wichtig sind, eingegangen werden. Zuvor werden kurz die Schwankungen vorgestellt, die für die Bereiche der Grob- und Mittelfristplanung bedeutend sind. Allerdings überlappen sich auch hier die Bereiche, so daß eine klare Trennung, welche Schwankungen welchen Bereich betreffen, nicht möglich ist.

In der Grobplanung steht die Abschätzung des Marktverhaltens im Mittelpunkt. Die Auftragseingänge, die ein Unternehmen erhält, sind v.a. bei Auftragsfertigern nicht im voraus bekannt und unterliegen zeitlichen Schwankungen. Daher wird in der Grobplanung versucht, das Verhalten des Marktes in langfristige Trends, saisonale Schwankungen und Strukturbrüche zu zerlegen und darauf aufbauend die Kapazitäten und die Beschaffung von Materialien anzupassen. Zur Erstellung der Prognosen müssen Termin und Umfang von Auftragseingängen und die Beschaffungszeiträume für Materialien statistisch untersucht werden. Die üblichsten Verfahren wie Regressionsverfahren und exponentielle Glättung erläutert [Ker94].

Die Mittelfristdisposition muß für die Materialdisposition Schwankungen im Bedarf, Verbrauch und Bestand berücksichtigen. Diese Unsicherheiten haben u.a. folgende Ursachen:

- **Fehler in der Bestandsführung:** Die im Datenbanksystem und in der Realität vorhandenen Bestände stimmen oft nicht überein. Dies liegt an ungenauen oder falschen Eingaben innerhalb der Lagerhaltung oder Fehlern in Stücklisten. Zufällige Schwankungen dieser Art lassen sich nicht ausschließen, sondern nur durch organisatorische Maßnahmen (z.B. keine Materialentnahme ohne Buchung und Beleg) minimieren.
- **Differenzen zwischen grobgeplanten und tatsächlichen Verbräuchen:** Dabei spielt der Ausschuß während der Produktion, der Verbrauch von Ersatzteilen für beschädigte Maschinen sowie die Lieferung von Endprodukten eine Rolle.
- **Lieferterminabweichungen für Einkaufsteile:** Fehlt ein bestimmtes Einkaufsteil, so kann ein Auftrag möglicherweise nicht begonnen werden. Mit den Lieferanten bestehen zwar feste Liefertermine, doch deren Einhaltung kann nicht als gesichert betrachtet werden. Für die Berück-

sichtigung dieser Zufälligkeiten wird die Lieferbereitschaft der Lieferanten für ein bestimmtes Einkaufsteil aus den vorhandenen Lieferdaten geschätzt.

- **Liefermengenabweichungen** treten weniger bei Einkaufsteilen als bei Eigenfertigungsteilen auf, die durch Ausschuß in der Produktion schwanken können.

Zur Berücksichtigung dieser Schwankungen werden Sicherheitsbestände und -zeiten berechnet, die in die Planung miteingehen. Terminabweichungen für die Liefertermine werden durch Sicherheitszeiten aufgefangen, die aus den Soll- und Ist-Terminen der Liefertermine berechnet werden. Die Stücklieferbereitschaft berechnet die Wahrscheinlichkeit, daß ein bestimmtes Eigenfertigungsteil auch in ausreichender Menge im Zwischenlager vorhanden ist. Da in der Fertigung an vielen Stellen Ausschuß produziert werden kann, muß ein bestimmter Sicherheitsbestand vorhanden sein, der den Ausschuß auffangen kann. Die Auftragslieferbereitschaft kalkuliert die Wahrscheinlichkeit, einen Auftrag auch zu einem bestimmten Termin fertigzustellen. In [Ker94] wird eine ausführliche Beschreibung der Formeln zur Berechnung von Sicherheitszeiten, Stück- und Auftragslieferbereitschaft gegeben.

Innerhalb der Produktion kommt es zu einer Vielzahl zufälliger Schwankungen, die bei der Erstellung von Belegungsplänen mehr oder weniger genau berücksichtigt werden können. Im folgenden sollen einige der Zufallsquellen vorgestellt werden. Allerdings ist dabei zu beachten, daß diese Schwankungen sehr stark vom Fertigungsprozess abhängen, so daß kaum allgemein gültige Aussagen machbar sind. Es ist stets eine genaue Analyse des Fertigungsprozesses notwendig, um entscheiden zu können, welche Schwankungen überhaupt auftreten und wie stark sie sich bemerkbar machen. [Wit94] zählt als Beispiele für zufallsbedingte Einflüsse auf die Produktion folgende Punkte auf:

- Art und Umfang des Auftragseingangs in die Produktion
- Zeitdauer zwischen den Auftragseingängen
- Ausfall von Maschinen und Krankenstand des Personals
- Ausführungszeiten von Bearbeitung und Transport
- Auftreten von Qualitätsmängeln nach Art und Häufigkeit

In diese Liste könnte man noch Ereignisse aufnehmen, die von außen in die Fertigung eingehen können. So können zum Beispiel Streiks die Produktion zum Stillstand bringen.

Für den Bereich der Kurzfristdisposition sind v.a. diejenigen Zufallsereignisse bedeutend, die entweder innerhalb der Produktion auftreten oder den Umfang der Produktion beeinflussen. Da von Fall zu Fall unterschieden werden muß, welche Ereignisse relevant sind, sollen an dieser Stelle nur Beispiele vorgestellt werden:

- **Kurzfristiges Einplanen von Sonderaufträgen:** Wenn es vorkommen kann, daß sehr kurzfristig neue und dringliche Aufträge eingeplant werden müssen, können diese durch Sicherheitsbestände aufgefangen werden, z.B. indem nur ein Teil der voraussichtlich verfügbaren Kapazität verplant wird. Muß dann ein weiterer Auftrag eingeplant werden, kann dieser die vorhandenen Puffer aufbrauchen und so in den Belegungsplan hereingenommen werden.
- **Ausfall von Maschinen:** Maschinen unterliegen als technische Geräte einem gewissen Verschleiß. Deshalb kann bei fast allen Maschinen eine Störung auftreten, die durch Reparatur behoben werden muß. Je nach der Art der Maschine kann es dabei vorkommen, daß sie nur während einer Bearbeitung oder auch im Stillstand ausfallen kann oder daß bei der Störung gleichzeitig Ausschuß produziert wird. Auch die Reparatur der Maschine ist von Fall zu Fall verschieden. Es kann sein, daß die Reparatur durch einfache Handgriffe in kurzer Zeit erfolgen kann, oder daß verschiedene Ersatzteile und Servicetechniker benötigt werden, die wiederum beschränkte Ressourcen darstellen. Die Ausfallwahrscheinlichkeit der Maschine kann vom Zeitpunkt der letzten Wartung, von der Qualität des bearbeiteten Materials oder von der Qualifikation des Bedieners abhängen. Wie aus diesen Beispielen ersichtlich ist, muß für jede Maschine individuell ein Modell erstellt werden, das die Zufälligkeiten der Maschine qualitativ und quantitativ widerspiegelt.

- **Verschleiß von Werkzeug:** Ebenso wie Maschinen unterliegen Werkzeuge einem Verschleiß, weshalb ähnliche Störungen wie bei Maschinen auftreten können. Zusätzlich können sich Werkzeuge abnutzen, d.h. sie verlieren an Effektivität. Bei zu großer Abnutzung müssen sie ausgetauscht werden, was i.d.R. routinemäßig nach einer bestimmten Anzahl von Betriebsstunden erfolgt.
- **Krankheit von Personal:** Der durchschnittliche Krankenstand im Betrieb wird in vielen Unternehmen gemessen, meist um einen Eindruck von der Motivation der Mitarbeiter[6] zu bekommen. Für die Disposition der Ressource Personal ist dabei noch interessant, daß Krankheitswahrscheinlichkeit und Krankheitsdauer recht eigene statistische Verteilungen aufweisen, die vom gesellschaftlichen Umfeld abhängen. Die Krankheitsdauer eines Mitarbeiters hängt z.B. stark vom Wochentag ab, an dem er krank wird, da oft bis zum Ende der Woche krankgeschrieben wird. Montage und Freitage sind Tage, an denen Mitarbeiter öfter krankgeschrieben sind. Ebenso spielt die Lage von Feiertagen eine Rolle.
- **Ausführungszeiten von Bearbeitung und Transport** können unterschiedlich ausfallen. Das kann daran liegen, daß Teile manuell nachbearbeitet werden müssen oder daß aus Versehen ein bearbeitetes Teil nicht weitertransportiert wird. Bei gewissen technologischen Prozessen hängt die Bearbeitungsdauer vom Material ab, z.B. in der Halbleiterfertigung. Diese Dauern lassen sich oft durch Normal- oder Exponentialverteilungen beschreiben.
- **Ausschuß:** Während der Bearbeitung kann es vorkommen, daß das bearbeitete Teil zerstört wird und deshalb als Ausschuß nicht mehr weiterverwendet werden kann. Ausschuß kann beim Ausfall von Maschinen, durch unsachgemäße Bedienung oder auch durch die Fertigungstechnologie bedingt entstehen. Bei gewissen Arbeitsgängen können diese Teile nachbearbeitet werden, was die Bearbeitungszeit erhöht, oder müssen als Verlust abgeschrieben werden. Ausschuß muß in der Materialdisposition mitberücksichtigt werden, da fehlende Eigenfertigungsteile einen ganzen Auftrag blockieren können. Daher ist hier ein bestimmter Sicherheitsbestand notwendig.

Die Beispiele zeigen, daß es wichtig ist, den Fertigungsprozess detailliert zu modellieren, um qualitative und quantitative Aussagen machen zu können. Dabei muß zuerst festgestellt werden, welche zufälligen Störgrößen relevant sind. Diese sollten statistisch aussagekräftig gemessen werden und aus den Messwerten die Sicherheitsgrößen abgeleitet werden. Die Messung und Feststellung der Störgrößen ist eine Systemanalyse (s. Abschnitt 2.4), die bei der Erstellung von Simulationsmodellen erläutert werden. Die Herleitung der Sicherheitsgrößen kann durch analytische Mittel, z.B. aus der Statistik, oder durch eine Simulationsstudie erfolgen.

4.6. Bewertungskriterien für dynamische Eigenschaften von Belegungsplänen

Im vorigen Abschnitt wurde veranschaulicht, daß der Fertigungsprozess einer Vielzahl von zufälligen, nicht vorhersehbaren Schwankungen unterliegt. Daher wird in der Realität kaum ein Belegungsplan exakt eingehalten. Je weiter der Zeitpunkt der Planerstellung zurückliegt, so größer werden die Abweichungen zwischen Plan und Wirklichkeit. Dieses Problem wird dadurch behoben, daß in regelmäßigen Abständen eine Um- oder Neuplanung durchgeführt wird, um einen an die aktuellen Umstände angepassten Plan zu erhalten. Je nach Art der Fertigung und der Leistung des eingesetzten PPS-Systems wird heutzutage eine regelmäßige Umplanung auf täglicher oder wöchentlicher Basis durchgeführt. Das Problem, daß erzeugte Pläne schon zu ihrer Erstellungszeit so veraltet und fehlerbehaftet sind, daß sie von den Betroffenen weitgehend ignoriert und durch ihr eigenes, lokal beschränktes Erfahrungswissen ersetzt werden, ist eher ein Problem der Betriebsdatenerfassung und Betriebsorganisation und soll an dieser Stelle nicht betrachtet werden. Ebenso verursacht das Erstellen von Belegungsplänen Kosten, so daß sich Einsparungen ergeben, wenn Umplanungen nur dann durchgeführt werden, wenn sie „notwendig“ sind.

[6] Oder einer Mitarbeiterin. Im folgenden beziehen sich die Worte „Mitarbeiter“, „Werker“ etc. auf erwachsene Menschen beiderlei Geschlechts.

Um zu erkennen, wann eine Umplanung notwendig ist, können die in Abschnitt 4.4 vorgestellten statischen Bewertungskriterien für Belegungspläne keinen Beitrag leisten, da von ihnen die vom zeitlichen Verlauf abhängige Umsetzung des Planes in die Realität nicht berücksichtigt wird. Daher werden in diesem Abschnitt Bewertungskriterien für Belegungspläne vorgestellt, die diese zeitliche Dynamik berücksichtigen. Dabei kann die Umsetzung des Belegungsplanes sowohl in der realen Fertigung als auch in einem Simulationssystem zur Verbesserung der Disposition erfolgen.

Durch den tatsächlichen Ablauf in der Fertigung entsteht im Laufe der Zeit ein eigener Belegungsplan, der mehr oder weniger große Ähnlichkeiten mit dem zuvor erstellten Belegungsplan besitzt. Daher bietet es sich an, dynamische Bewertungskriterien für Belegungspläne auf dem Vergleich zwischen diesen beiden Pläne aufzubauen. Der Vergleich zwischen den beiden Plänen erfolgt auf der Grundlage, daß die beiden Ausgangsmengen der Zuordnung (Nachfrager und Anbieter) identisch sind und nur bestimmte Eigenschaften der Objekte und die getroffene Zuordnung unterschiedlich sein können. Für Ressourcenbelegungspläne können die beiden Pläne sich z.B. in Bezug auf Bearbeitungszeiten für Operationen, benötigte Materialien und beanspruchte Ressourcen unterscheiden. Ebenso kann sich die Reihenfolge, in der Operationen auf einer Ressource bearbeitet werden, ändern.

Zum Vergleich zwischen Plan und Ablauf kann im einfachen Fall die aktuelle Lage des Ablaufs zu einem Zeitpunkt betrachtet und für diesen Zeitpunkt eine Bewertung durch die Unterschiede zwischen beiden durchgeführt werden. Dabei wird das Geschehen in der Vergangenheit, das zu dem aktuellen Zustand geführt hat, nicht berücksichtigt. Im komplexen Fall wird nicht nur der aktuelle Zustand, sondern auch der Verlauf, der zum aktuellen Stand geführt hat, betrachtet. Als Zeitraum für den Verlauf kann ein konstantes Intervall oder der gesamte Planungszeitraum bis zum aktuellen Stand dienen. Dabei können die beiden Pläne getrennt bewertet und anschließend die Bewertungsergebnisse miteinander verglichen werden. Auf diese Weise können Bewertungskriterien, die für die Erstellung von Plänen angewendet werden, für den Vergleich zweier Pläne verwendet werden. Da die üblichen Bewertungskriterien für die Erstellung von Belegungsplänen allerdings nur die statischen Eigenschaften des Planes beurteilen, entstehen somit Bewertungsmaße, deren dynamische Eigenschaften nur daher stammen, daß sie während des Ablaufs mit jeweils erweiterten Belegungsplänen arbeiten.

In der Literatur sind wenige Arbeiten zum Vergleich von Belegungsplänen und zur Bewertung der dynamischen Eigenschaften von Schedules zu finden. Ein Ansatz, der auf Fließfertigungsstrukturen innerhalb der Elektronikproduktion aufsetzt, ist in [Sau95] zu finden. Dabei geht der Ansatz im besonderen davon aus, daß innerhalb eines Auftrags die Reihenfolge der Operationen nicht geändert werden darf. [Sau95] stellt zwei Maße zum Vergleich zwischen Ablauf und Plan vor, die nur den aktuellen Zustand des Systems und den Ablaufplan benötigen:

- ***Pass Error:*** mißt die Anzahl der verspäteten Operationen zu einem bestimmten Zeitpunkt.

  1. Für jede Maschine wird die gerade bearbeitete Operation O_1 betrachtet.
  2. Es wird im Plan nachgeschaut, welche Operation O_2 gerade durchgeführt werden sollte.
  3. Es werden alle Operationen ermittelt, die vor O_2 noch bearbeitet werden müssen. Das sind die zwischen O_1 und O_2 auf der Maschine angesetzten Operationen.
  4. Diese Anzahl noch auszuführender Operationen wird für jede Maschine ermittelt und aufsummiert. Ist die Ressource dem Plan voraus, so erhält man ein positives Vorzeichen, bei Verspätung ein negatives. Das Ergebnis wird durch die Anzahl aller Operationen geteilt. Dadurch erhält man eine Bewertung, die nicht mehr von Umfang und Anzahl der Aufträge abhängt.
  5. Überschreitet das Ergebnis eine gewisse Schranke (z.B. $\pm 3\%$), so wird eine Umplanung durchgeführt. Dabei kann die Schranke sowohl nach oben als auch nach unten überschritten werden, da die Verfrühung von Operationen eine negative Bewertung ergibt.

  Für jede Maschine wird die gerade bearbeitete Operation O_1 betrachtet. Wenn Operationen auf einer Maschine stark unterschiedliche Bearbeitungsdauern haben, so bewertet das obige Kriteri-

um kürzere Operationen stärker als längere, da die Bearbeitungsdauer nicht in das Kriterium eingeht.

- ***Duration Error:*** mißt die Dauer der Verspätung. Der Algorithmus ist analog zum Pass Error bis auf:

4a. Für jede der noch auszuführenden Operationen wird die (geschätzte) Bearbeitungszeit ermittelt und aufsummiert. Das Ergebnis wird durch die Summe der Bearbeitungszeiten aller Operationen geteilt.

In Abschnitt 5.4 wird das Simulationssystem ROSI vorgestellt und eine Umplanung (dort „Synchronisation“ genannt) anhand dieser beiden Kriterien durchführt. Die Überprüfung erfolgt in ROSI sowohl aufgrund des Zustands von Operationen oder Ressourcen (z.B. bei einer Störung) als auch periodisch. Die Maße sind einfach zu implementieren und effizient zu überprüfen.

Für den Grundalgorithmus von [Sau95] sind viele Erweiterungen denkbar. Eine dynamische Bewertung der Termintreue kann z.B. erreicht werden, in dem man die Entfernung zum spätesten Termin der Operation schätzt. Während in den ersten beiden Kriterien die Verspätung berücksichtigt wird, geht in diese Bewertung der noch vorhandene Puffer ein. Solange noch ausreichend Puffer vorhanden ist, sind Verspätungen als solche unproblematisch.

- ***Restpuffer:*** mißt die noch vorhandenen Puffer.

4b. Für jede der noch auszuführenden Operationen wird die (geschätzte) Bearbeitungszeit ermittelt und ein neuer geschätzter Plantermin durch Hochrechnung des aktuellen Standes ermittelt. Zu dem so errechneten neuesten Endtermin der Operation wird ihr Abstand zum spätesten Endtermin (SET) ermittelt. Diese Differenzen werden aufsummiert, wobei eine Überschreitung des SET zusätzlich stärker gewichtet werden können. Das Ergebnis wird durch die Summe der für diese Operationen bereitgestellten Pufferzeiten zwischen Endtermin und SET geteilt.

5b. Die Umplanung wird beim Unterschreiten einer Schranke (z.B. *<3%*) ausgelöst.

Zur Bewertung der Auslastung können Maße konstruiert werden, die z.B. den Anteil der wartenden (d.h. freigegebenen, aber nicht in Bearbeitung befindlichen) Operationen und der untätigen Ressourcen miteinander in Verbindung setzen.

Welches Maß zur Bewertung benutzt werden sollte, hängt aber stark von den Eigenschaften des Planes (und somit von Disponent und Planerstellungssoftware), den Fertigungsstrukturen und den Zielen eines Betriebes ab, so daß diese Entscheidung von Fall zu Fall getroffen werden sollte. In DYBBS (s. Abschnitt 6.3.5) ist zum Auslösen einer Umplanung ein Mechanismus eingebaut, der es erlaubt, rein situationsbezogene Bewertungskriterien der vorgestellten Bauart einfach zu implementieren, so daß sie vom Experten angewendet und parametrisiert werden können. In DYBBS sind die folgenden dynamischen Bewertungsfunktionen implementiert:

- Anzahl arbeitender bzw. gestörter Ressourcen
- Anzahl wartender Operationen, Dauer der wartenden Operationen
- Pass Error, Duration Error, Restpuffer (s.o.)

Um ein Bewertungsmaß für die Auslösung einer Umplanung zu implementieren, das den Verlauf berücksichtigt, muß im Prinzip eine Trendvorhersage durchgeführt werden, die vom bisherigen auf den zukünftigen Verlauf schließt. In DYBBS wurde kein Bewertungskriterium implementiert, das den Verlauf zur Bewertung berücksichtigt. Die Möglichkeit, solche Bewertungsmaße zu implementieren, ist allerdings vorhanden, da der Ablauf der Simulation in eigenen Belegungsplänen dargestellt wird, die zur Bewertung ausgewertet werden können. Für die Gesamtbewertung eines Simulationslaufes können hingegen diese Belegung und der zugrundeliegende Belegungsplan durchaus durch die Bewertung aufgrund der statischen Bewertungsmaße verglichen werden.

Abbildung 12 zeigt ein einfaches Beispiel für Schedule und Ablauf an einer Maschine. Insgesamt sind 10 Operationen eingeplant. Zum Betrachtungszeitpunkt t_2 ist gerade Operation 6 beendet worden, wohingegen im Plan bereits das Ende von Operation 9 vorgesehen war. Auf dieses einfache Beispiel werden die vorgestellten Bewertungskriterien angewendet (s. Tabelle 4).

	t_1	t_2
Pass Error	0	$\frac{3}{10} = 30\%$
Duration Error	0	$\frac{1+1+1}{22} \approx 13{,}6\%$
Restpuffer	$\frac{17}{18} \approx 94\%$	$\frac{-2-1-1-1-1}{2+2+2+2+1} \approx -67\%$

Tabelle 4: Auswertung der Bewertungskriterien zu Abbildung 12

Es zeigt sich, daß zum Zeitpunkt t_1 die Abweichung als unbedeutend bewertet wird. Zum Zeitpunkt t_2 schlagen die Bewertung „Pass Error“ und „Restpuffer“ sehr hoch aus, während die Bewertung „Duration Error“ die Situation weniger stark bewertet, da die verspäteten Operationen kurze Dauer haben.

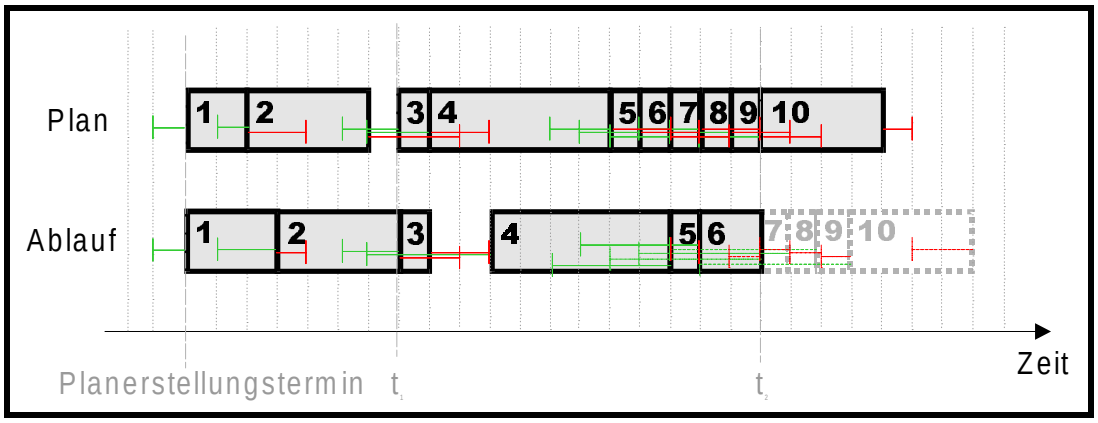


Abbildung 12: Vergleich zwischen Plan und Ablauf

4.7. Zusammenfassung

Die in diesem Kapitel ausgeführten Zusammenhänge zeigen, daß verschiedene Einsatzgebiete für Simulation innerhalb von PPS-Systemen existieren. Simulation kann dabei sowohl zur Erstellung als auch zur Bewertung von Belegungsplänen verwendet werden. Während bei der Erstellung von Belegungsplänen verschiedene Ansätze miteinander konkurrieren, kann die Bewertung der dynamischen Eigenschaften von Belegungsplänen nur durch eine entsprechende Simulation durchgeführt werden, da in den anderen Ansätzen die dynamischen Eigenschaften des Fertigungsprozesses nur unzureichend berücksichtigt sind. Durch die Möglichkeit, den Fertigungsprozess eines Betriebes oder einen Belegungsplan unter einer speziellen Fragestellung zu simulieren, ist Simulation ein wichtiges Hilfsmittel für PPS-Systeme.

Für ein Simulationswerkzeug für PPS-Systeme ist es wichtig, dem Modellierer einen einfachen Zugriff auf die für die Simulation wichtigen Daten des PPS-Systems zu geben und die Modelle trotz der Vielzahl von vorhandenen Objekten und Daten übersichtlich und intuitiv zu gestalten. Für das erste Ziel ist somit eine Integration in ein PPS-System notwendig, während für die zweite Aufgabe eine dem Problem angepasste Modellierungsmethode gefunden werden muß. Um die vielfältigen Fragestellungen zu berücksichtigen, unter der eine Simulation eines Fertigungssystems stattfinden kann, muß das Simulationswerkzeug leicht erweiterbar sein. Daher steht für die Entwicklung einer solchen Simulationsshell nicht die Implementierung spezieller Bewertungskriterien im Vordergrund, sondern es wird eine ausbaufähige Architektur benötigt, in der solche Bewertungsfunktionen einfach implementiert und auf ein Simulationsmodell angewendet werden können.

5. Modellierung einer auftragsgesteuerten Fertigung in Simulatoren

In diesem Abschnitt sollen drei Simulationstools - ARENA, SIMPLE++ und ROSI - vorgestellt werden, mit denen eine Simulation auftragsorientierter Produktionsprozesse möglich ist. Wegen der Beschränkung auf eine auftragsorientierte Fertigungssimulation werden an dieser Stelle die Vielzahl von Simulationstools, die sich mit physikalischen, geographischen, biologischen Systemen oder mit der Modellierung von Multiagentensystemen befassen, nicht betrachtet. Die im Rahmen dieser Arbeit implementierte Simulationsshell DYBBS wird in Kapitel 6 erläutert. Die betrachteten Tools arbeiten alle nach der Architektur einer DES, die in Abschnitt 2.3 beschrieben ist.

Der Schwerpunkt bei der Untersuchung der Simulationstools wurde auf deren Möglichkeiten zur Erstellung eines auftragsorientierten Produktionsmodells gelegt. Insbesondere auf die Simulation eines Belegungsplanes und die Verwirklichung von Umplanungen während eines Simulationslaufes soll eingegangen werden.

5.1. Klassifikation von Simulationssoftware

Generell läßt sich Simulationssoftware in die Klassen Simulationssprachen und Simulatoren (Simulationstools) unterteilen ([Law91], S. 236ff). Eine Simulationssprache ist allgemein gehalten, so daß eine große Menge von Systemen mit ihr modelliert werden können. Dazu benötigt der Anwender aber Programmierkenntnisse. Ein Simulator ist auf eine Systemklasse spezialisiert und erlaubt für diesen Typ eine einfache, schnelle und graphische Modellierung. Zusätzlich gibt es Mischformen aus beiden Typen, die versuchen, die Stärken beider Klassen zu vereinen. Ein Beispiel dafür ist das im folgenden Abschnitt vorgestellte ARENA. Eine ausführlichere Übersicht über klassische Simulationssprachen, z.B. GPSS/H, SIMSCRIPT II oder SLAM II, befindet sich in [Law91]. [Wit94] gibt eine Einführung in SLAM II mit besonderer Berücksichtigung der Simulation von Fertigungsprozessen.

5.2. ARENA

5.2.1. Einführung

ARENA stellt eine Weiterentwicklung der Simulationssprache SIMAN (Simulation Analysis) von Dennis Pegden dar, die seit 1982 existiert. ARENA ist ein kommerzielles Simulationspaket von Systems Modelling Inc.[7], das seinen Schwerpunkt auf die Simulation von Produktionsprozessen legt. Vorgestellt wird die Version 3.0 (eingeschränkte Studentenversion) unter dem Betriebssystem Windows 95/NT4.0. Durch seine Vielzahl vorhandener Komponenten und die Schnittstellen zu Excel (für Daten) und VisualBasic[8] (für eigene Programme) ist es für alle Simulationen, die sich mit einem diskreten ereignisorientierten Simulationsansatz sinnvoll verwirklichen lassen, geeignet.

In ARENA erfolgt die Modellierung eines Systems durch das Plazieren und Verbinden von Elementen, die aus einer Modellbibliothek auf das Modellierungsfenster gebracht werden. Als Elemente dienen dabei komplexe Strukturen wie ***Ankunftsquellen, Bedieneinheiten, Warteschlangen, Transportvorrichtungen***, usw. Allerdings geht ARENA in diesem graphischen Ansatz weiter, da für alle Strukturen der zugrundeliegenden Simulationssprache SIMAN, also z.B. für Fallunterscheidungen oder Zuweisungen, ein graphisches Element geschaffen wurde. Die komplexen Strukturen bestehen im Prinzip aus einer Gruppierung solcher Sprachelemente. Die einzelnen Elemente werden in ihrer Durchlaufreihenfolge miteinander verbunden, was insbesondere die Modellierung einer Fließfertigung sehr einfach macht. Das Modellieren in ARENA hat somit gewisse Ähnlichkeiten mit der Definition von Flußgraphen für strukturierte Programme.

[7] Systems Modelling Inc.: http://www.sm.com/
[8] Microsoft Gmbh.: http://www.microsoft.de/

Grundsätzlich besteht ein Modell in ARENA aus einer Menge von ***Quellen (Arrive)***, ***Bedieneinheiten (Server), Senken (Depart)*** und einem ***Simulatorelement (Simulate)***. Arrivals erzeugen die zu bearbeitenden Anforderungen. Dies kann aufgrund einer Zufallsverteilung oder durch das programmgesteuerte (VisualBasic) Auslesen einer Tabelle (aus Excel) erfolgen. Einer Anforderung kann eine Menge von Attributen zugewiesen werden, wobei immer ein Zeitstempel-Attribut vorhanden ist. Nach der Erzeugung einer Anforderung wird diese zu einem Server geleitet. Welcher Server ausgewählt wird, hängt von der Programmlogik ab, die nach der Quelle eingebaut ist. Am Server wird die Anforderung in eine Warteschlange eingereiht. Wenn die Anforderung bearbeitet werden kann, wird sie aus der Warteschlange entfernt und in den Server aufgenommen. Nach Beendigung der Bedienung erfolgt ein Weitertransponsport zu einem anderen Server oder zu einer Departure. Bei einer Departure wird die Anforderung aus dem System entfernt. In Arena wird somit die eigentliche Ereignismodellierung in den komplexen Elementen durch eine transaktionsorientierte Modellierung verborgen. Das Simulatorelement steuert die Dauer der Simulation und die Animationen. Nach Beendigung der Simulation wird eine statistische Analyse präsentiert.

Abbildung 13 zeigt ein einfaches Beispiel. Auf einem Förderband kommen zufällig Teile an. Es gibt zwei Bedieneinheiten. Die Entscheidung, welche Bedienstation gewählt wird, wird aufgrund der Anzahl der vor den beiden Servern wartenden Einheiten getroffen. Fertige Teile werden über ein zweites Förderband aus dem System transportiert. Die Ankunft von Anforderungen und die Auswahl der Bedieneinheit erfolgen in den Elementen Arrive und Pickstation. Die Förderbänder sind durch die Verbindungen zwischen Pickstation, den Servern und dem Element Depart angedeutet.

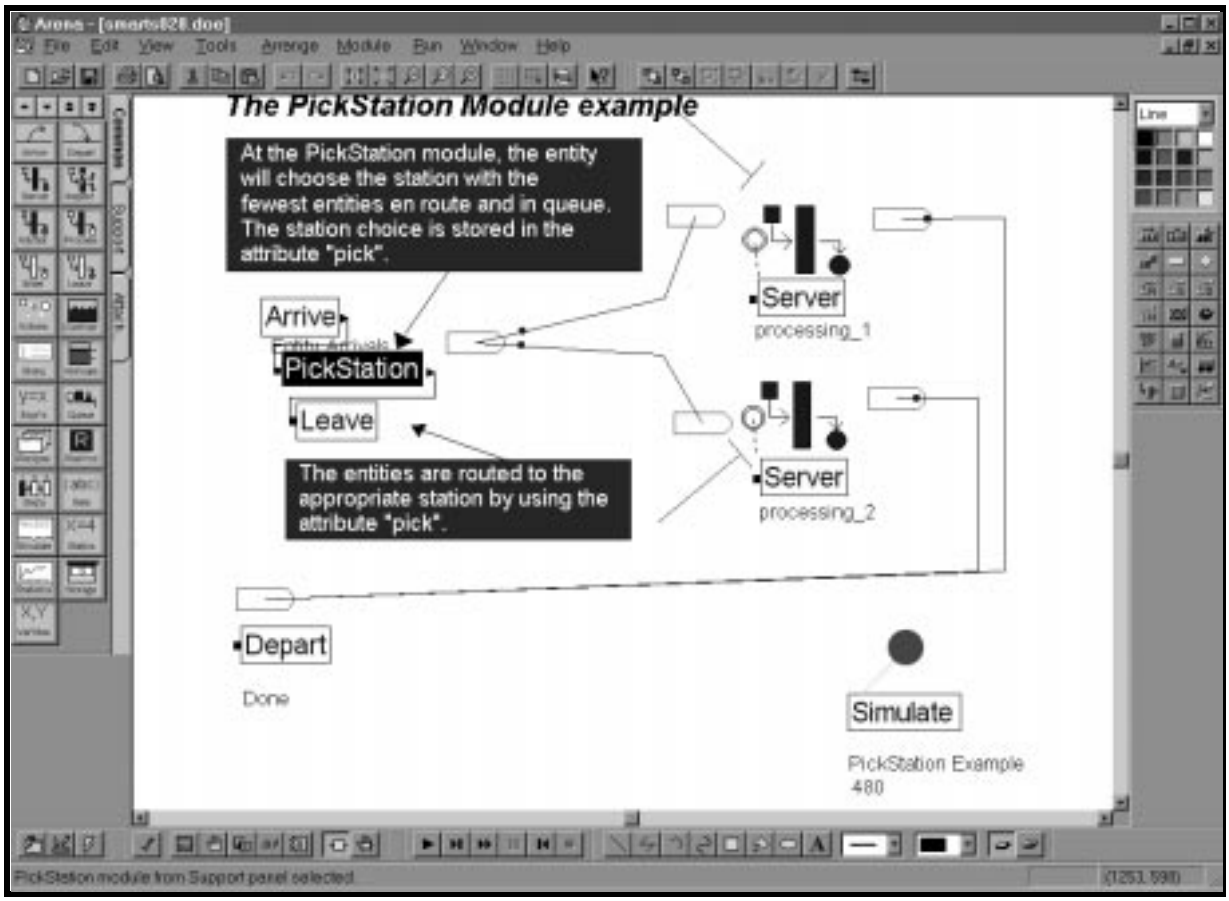


Abbildung 13: Beispiel aus der ARENA-Modellbibliothek

Die Vorteile von ARENA liegen zum einen in der Bereitstellung komplexer Elemente, die bereits für eine große Menge von Simulationstudien ausreicht, da sie sehr viele Dinge berücksichtigen. Abbildung 14 zeigt als Beispiel die vorhandenen Einstellungen für einen Server, in dem u.a. Kapazität (capacity), Bearbeitungszeit (process time), Ausfallhäufigkeit, Ausfalldauer (Options) und Transportdauer (route time) definiert werden können. Zum anderen ist durch die vielen primitiven Programmelemente eine

beliebige Flexibilisierung möglich. Dazu bietet ARENA eine hohe Performanz, die üblichsten Verteilungsfunktionen sowie eine statistische Vor- und Nachbereitung von Ein- und Ausgabe durch einen Input- bzw. Output-Analyzer.

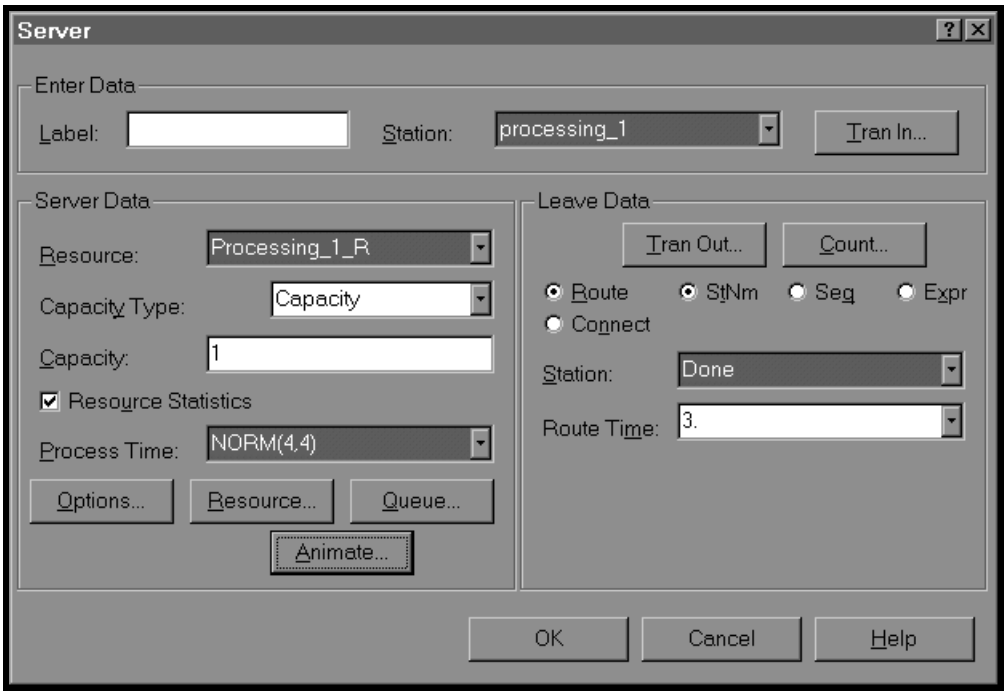


Abbildung 14: ARENA, Servereigenschaften

5.2.2. Simulation eines auftragsgesteuerten Fertigungssystems in ARENA

Bei der Vorstellung eines Simulationsmodells für eine aufragsorientierte Fertigungssteuerung sollen jedoch nur die Prinzipien erläutert werden, nicht jedoch eine komplette Implementierung. Für Hinweise, wie spezielle Einzelheiten implementiert werden müssen, steht in ARENA eine mitgelieferte Bibliothek von Beispiel-Modellen (Smarts-Library) zur Verfügung, die aus kleinen tutoriellen Modellen besteht und Implementierungstechniken aufzeigt.

Als Eingabe in ARENA wird ein Ressourcenbelegungsplan mit der Zuordnung von Aufträgen und Operationen auf die Ressourcen Arbeitsplätzen und Zeit vorausgesetzt (s. Abschnitt 4.2), der in Form einzelner Excel-Tabellen vorliegt. In diesem Beispiel wird weiterhin von einer bekannten und festen Menge von Arbeitsplätzen ausgegangen, damit diese statisch in das Modell eingebaut werden können. Der Einfachheit halber soll davon ausgegangen werden, daß jede Operation maximal einen Vorgänger und Nachfolger besitzen kann. Ebenso sind weder Werkzeuge, Material noch Personal berücksichtigt, da dieses die Komplexität dieses Beispiels unnötig erhöht, ohne neue Ergebnisse zu zeigen.

Für jeden Arbeitsplatz wird ein Server-Modul eingefügt, dazu jeweils ein Arrive-, ein Depart- und ein Simulate-Modul. Jeder Arbeitsplatz kann danach individuell eingestellt werden in Bezug auf Störungen und Eigenschaften der Bearbeitung (z.B. Verteilungsfunktion für Bearbeitungszeit). In der Quelle werden alle auszuführenden Operationen erzeugt und in einem VisualBasic-Modul mit Attributen über Bearbeitungszeit, zugeordnete Arbeitsplätze und Werker, Vorgänger- und Nachfolgeroperationen aus den Tabellen versehen. Nach dem Arrive-Modul folgt eine Wait-Bedingung, die auf ein Signal wartet. Das Signal wird jeweils nach der Beendigung einer Operation ausgelöst und hat als Inhalt die Nummer der beendeten Operation. Die Wartebedingung ist, daß das Attribut Vorgänger einer wartenden Operation auf das Signal der bearbeiteten Operation wartet. Operationen ohne Vorgänger umgehen diese Wartebedingung. Danach muß der zugeordnete Arbeitsplatz angesteuert werden. Dies erfolgt über eine Verzweigung (Choose) nach dem dafür vorgesehenen Attribut.

Im Server-Modul wird die Operation in eine Warteschlange eingereiht. Ist die Bedieneinheit verfügbar, so wird mit der Bearbeitung begonnen. Die Dauer der Bearbeitung wird aus der in der Operation vermerkten Dauer und der Verteilungsfunktion für diesen Server berechnet. Nach der Bearbeitung wird in einem Signal-Modul der Nachfolger der Operation freigegeben. Am Ende wird die bearbeitete Operation in einem Depart gelöscht, wobei Statistiken berechnet werden. Abbildung 15 zeigt den prinzipiellen Aufbau der so geschilderten ARENA-Implementation.

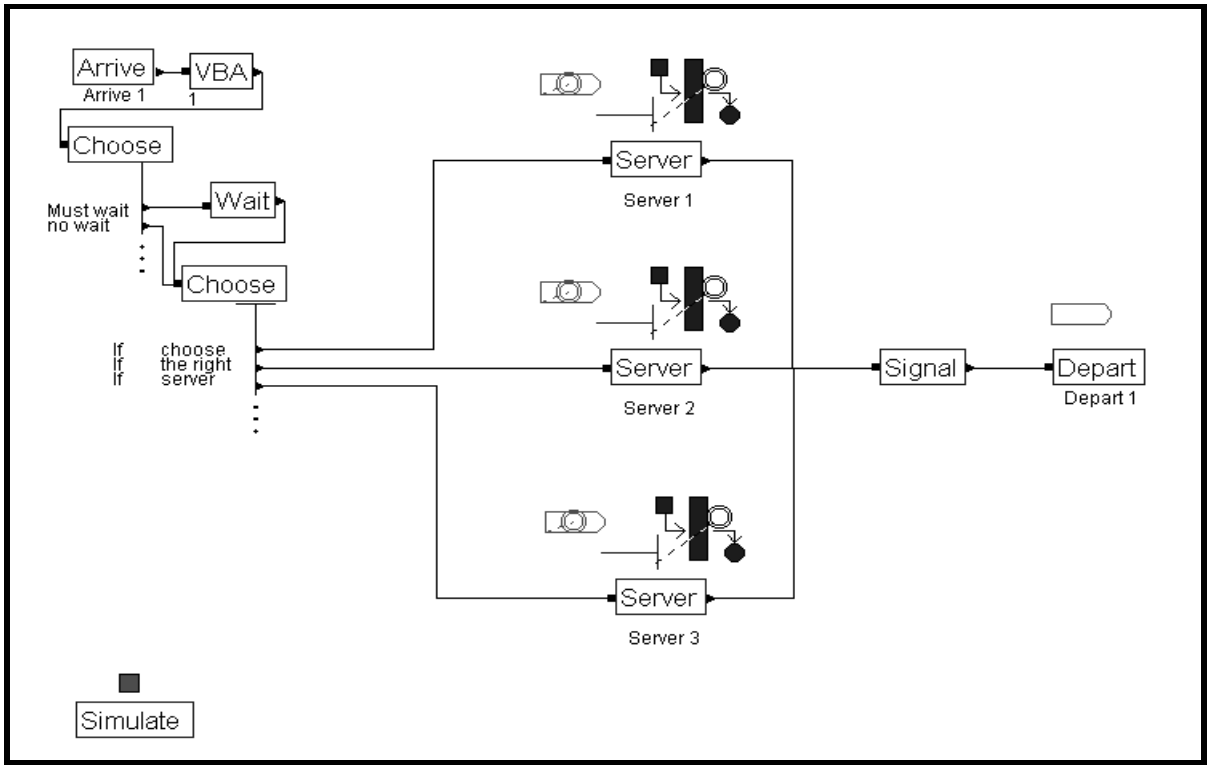


Abbildung 15: Auftragsgesteuerte Fertigung in ARENA

Als Erweiterung müßten in weiteren Schritten die Berücksichtigung von frühesten Terminen von Operationen und die Einbeziehung von Werkern, Material und Werkzeug erfolgen. Diese Einbeziehung ist im wesentlichen durch ein Duplizieren und Synchronisieren von Operationen zu erreichen. Ebenso müßten weitere VisualBasic-Funktionen für eine Erweiterung der Ausgabe geschrieben werden. Ebenso könnte über ein aufwendigeres Modul das komplette Simulationsmodell aus PPS-Daten erstellt werden, in dem zusätzlich zu den Aufträgen auch die vorhandenen Arbeitsplätze eingelesen und als Server im Modell hinzugefügt werden. Da sich allerdings diese Daten für eine Fertigungsumgebung sehr langsam ändern, kann das Modell im Bedarfsfall auch manuell ergänzt werden. Was hingegen schwierig zu verwirklichen ist, ist die Durchführung einer Umplanung. Diese müßte vollständig in VisualBasic erfolgen und gleichzeitig Methoden aus einem PPS-Werkzeug aufrufen. Damit müssen für diese Simulation dann vier Applikationen (ARENA, VisualBasic, Excel, PPS-Werkzeug) aktiv sein, die jeweils für Teilbereiche zuständig sind und miteinander synchronisiert werden müssen. Trotz der hohen Effizienz und der vielen vorhandenen Module in ARENA ist dies einer der Gründe, warum DYBBS als eigene Simulationsshell in Common-LISP und voll integriert in WIZARD implementiert wurde.

5.3. SiMPLE++

SIMPLE++ (Simulation in Produktion, Logistik und Engineering) ist ein Simulationswerkzeug für den Entwurf und die Optimierung von Fertigungsprozessen, das von der Firma Tecnomatix[9] vertrieben wird. In einer eigenen Einschätzung bezeichnet sich SIMPLE++ als (zur Zeit einziges) Simulationswerkzeug

[9] Tecnomatix: http://www.tecnomatix.com/ (früher Aesop Gmbh)

der 4. Generation.[10] SIMPLE++ zeichnet sich gegenüber anderen graphischen, bausteinorientierten Simulationswerkzeugen durch einen objektorientierten Ansatz und eine sehr hohen Integrationsfähigkeit in die Softwareumgebungen von Betrieben aus. SIMPLE++ ist in der Version 4 unter UNIX-Betriebssystemen und MS Windows NT4.0/95 erhältlich.[11]

Der Haupteinsatzbereich liegt in der Simulation von automatisierten Fertigungsprozessen, wie sie z.B. im Automobilbau vorliegen. Zusammen mit anderen Werkzeugen (ROBCAD, DYNAMO) kann mit SIMPLE++ aus den CAD-Daten von Produktbeschreibungen und Industrierobotern ein Entwurf einer einzelnen Fertigungszelle simuliert werden. Beispielsweise soll eine Fertigungszelle aus mehreren Industrierobotern einen Motorblock in die Karosserie einsetzen, wobei sich das Problem ergibt, daß die einzelnen Roboter sich nicht gegenseitig behindern dürfen und die einzelnen Arbeitsschritte in einer optimalen, a priori nicht bekannten Reihenfolge auszuführen sind.

Daraufhin kann das Verhalten der einzelnen Fertigungszellen in ihrem Zusammenspiel untersucht werden, wobei in SIMPLE++ eine Vielzahl von Bausteinen für den Transport von Material (z.B. Fließbänder, fahrerlose Transportsysteme) bereitstehen. Anhand der CAD-Daten über die Werkhalle können verschiedene Layoutanordnungen der einzelnen Fertigungszellen, der Pufferräume und ihrer Verbindungen simuliert werden, um eine kosten- und leistungsoptimale Lösung für den Entwurf des Fertigungssystems zu finden. In einem nächsten Schritt kann dann das Zusammenwirken einzelner Werkhallen mit dem An- und Abtransport von Material auf einem Werksgelände und sogar das Zusammenwirken mehrerer Werke an unterschiedlichen Standorten simuliert werden. Dieses Beispiel zeigt, daß in SIMPLE++ eine sehr tiefgreifende hierarchische und feine Modellierung möglich ist, bei der das Modell auf verschiedenen Detaillierungsebenen betrachtet werden kann.

Die Modellierung in SIMPLE++ erfolgt ähnlich wie in Arena über eine graphische Benutzeroberfläche, in der aus einer Bibliothek von Modellbausteinen Elemente ausgewählt, auf einer Modellierungsfläche plaziert und miteinander verbunden werden. Dabei kann als Grundlage der Modellierungsfläche ein CAD-Entwurf dienen, so daß Anordnung und Größe der verwendeten Elemente auch sichtbar mit den Vorgaben übereinstimmen. Dadurch wird z.B. die Modellierung von Materialtransportsystemen erheblich vereinfacht, da hier die Weglängen im Modell mit den realen Weglängen übereinstimmen und sich daraus automatisch die Transportdauern ergeben.

Die Objektorientierung von SIMPLE++ zeigt sich darin, daß z.B. eine Fertigungszelle eine Klasse ist, deren Verwendung in den einzelnen Modellen die jeweiligen Instanzen der Klasse darstellen, wobei verschiedene Attribute (z.B. die Anzahl der Roboter) festgelegt werden können. Für eine Klasse können in der Steuerungssprache SimTALK Methoden definiert werden, die das Verhalten bestimmen. Somit müssen ähnliche Komponenten nur einmal als eine Klasse aus einer Menge von Standardbausteinen entworfen werden und können wiederverwendet und spezialisiert werden. Die hierarchische Modellierung wird in SIMPLE++ dadurch erreicht, daß auch ein gesamtes Modell als eigene Klasse in den Bausteinkomponenten abgelegt und somit als Untermodell in ein anderes Modell eingebaut werden kann. Durch die Tatsache, daß ein komplettes Modell selbst eine Klasse ist, können mehrere Instanzen dieser Klasse angelegt werden, wodurch mehrere Simulationsmodelle parallel simuliert werden können. Durch die Integrationsmöglichkeiten über Datenbankschnittstellen und Prozeßkommunikation kann SIMPLE++ relativ einfach in das vorhandene PPS-System eines Betriebes eingebunden werden.

Durch das Zusatzmodul PROFIT kann ausgehend von einem PPS-System eine Simulation eines Fertigungsprozesses durchgeführt werden. PROFIT kann dabei zur Optimierung von Belegungsplänen durch Simulation (z.B. im Zusammenspiel mit genetischen Algorithmen), Schwachstellenanalyse oder zur Überwachung des Fertigungsprozesses durch einen Soll-Ist-Vergleich eingesetzt werden. Damit können in SIMPLE++ mit PROFIT durch das Zusammenführen von PPS-System und Simulation die in

[10] Die 1. Generation umfaßt allgemeine Programmiersprachen, die 2. Spezielle Simulationssprachen und die 3. Generation bausteinorientierte, graphische Simulationswerkzeuge.

[11] SiMPLE++ konnte leider nicht praktisch evaluiert werden. Die Evaluation fand anhand von Handbüchern einer älteren Version und der Teilnahme an einem Workshop bei der Firma Aesop statt.

Kapitel 4 beschriebenen Einsatzgebiete für Simulation in der Fertigung realisiert werden, wobei durch in diesem Fall viele Grundfunktionen durch die Module abgedeckt sind, während für spezielle Fragestellungen eigene Funktionen programmiert werden müssen. Insbesondere sind die Bereiche Scheduling durch Simulation, Fertigungsplanung und Berücksichtigung zufälliger Einflüsse auf den Fertigungsprozeß abgedeckt. Die in DYBBS realisierten Möglichkeiten der Bewertung von Belegungsplänen durch Simulation und der ereignisgesteuerten Umplanung sind zwar in SIMPLE++ nicht vorhanden, können dort aber in SimTALK implementiert werden.

5.4. ROSI

ROSI steht für „Reihenfolgeoptimierung durch Simulation“ und ist ein Simulationstool, das am Institut für Elektronik-Technologie der TU Dresden entwickelt wurde ([Sau96]). Es ist auf die auftragsgesteuerte Produktion (innerhalb der Elektrotechnik) spezialisiert, weshalb es an dieser Stelle untersucht wird. Die Spezialisierung auf das Gebiet der Elektrotechnik macht sich v.a. in der Terminologie des Programms bemerkbar, stellt aber ansonsten nur eine geringe Einschränkung dar. Das Tool läuft unter UNIX-Betriebsystemen mit der graphischen Oberfläche X-Windows/Motif und ist frei erhältlich.[12] Verwendet wurde die Version XROSI 1.7 unter dem Betriebssystem LINUX.

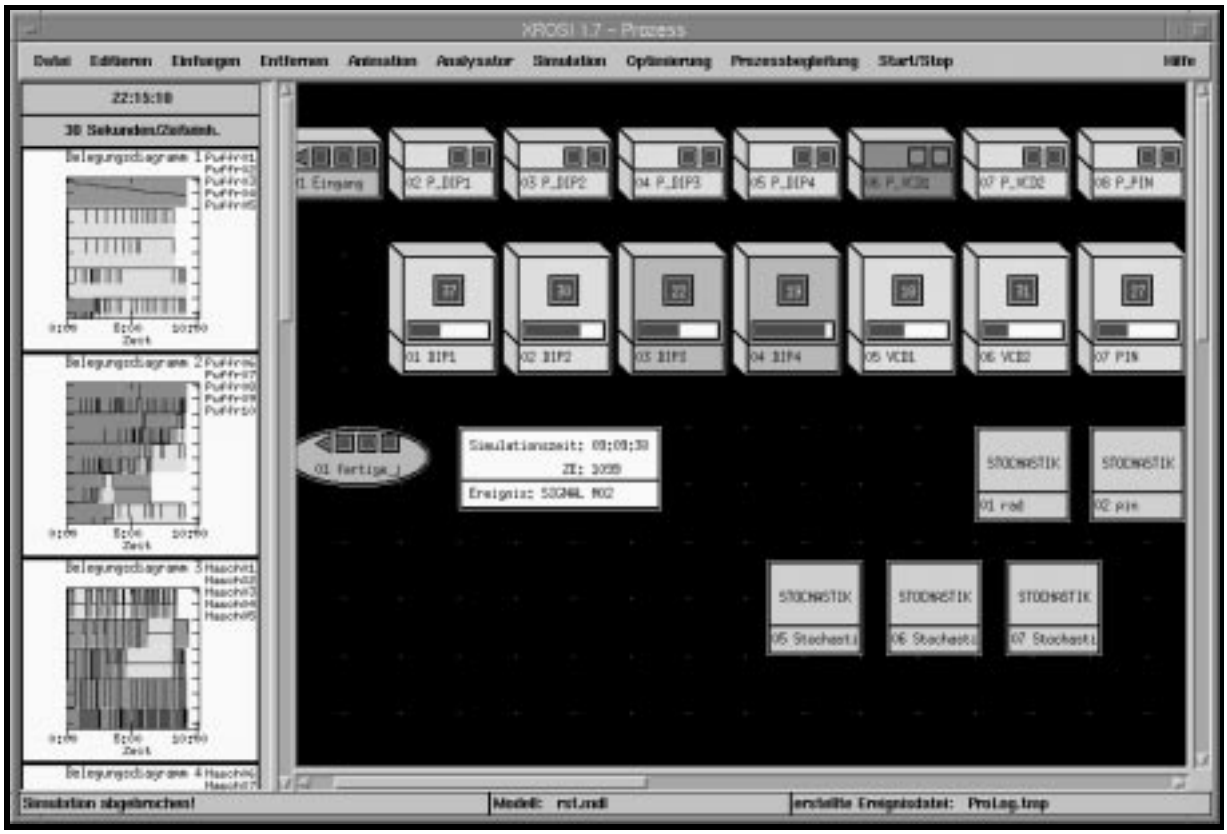


Abbildung 16: XROSI - Beispielmodell

Die Modellierung eines Fertigungssystems unter XROSI erfolgt ähnlich zur Vorgehensweise in ARENA. Zuerst wird ein Modell des zu modellierenden Produktionssystems erstellt, indem ***Bedieneinheiten, Pufferlager (Warteschlangen), Quellen*** und ***Senken*** per Drag&Drop auf der Modellierungsfläche verteilt werden. Danach werden die einzelnen Eigenschaften der definierten Elemente bestimmt. Dies sind z.B. in einer Quelle die zu bearbeitenden Aufträge oder die Bediendauer in einer Bedienstation. Die in einer Quelle entstehenden Aufträge können mit Prioritäten versehen werden, welche die Reihenfolge der Weitergabe bestimmen. Die physikalischen Zwänge, in welcher Reihenfolge Arbeitsgänge ausgeführt werden können (z.B. Bestücken vor Löten) werden in einer ***Technologie*** beschrieben. ***Aufträge*** stellen ein Los von Erzeugnissen dar, die hintereinander gefertigt werden. Ein Auftrag (d.h. ein

[12] http://www.et.tu-dresden.de/iet/ietproz-rosi.html

Erzeugnis in einer bestimmten Losgröße) kann nicht mehr aufgeteilt werden. In jedem Auftrag kann maximal ein Arbeitsgang zu einem Zeitpunkt aktiv sein, d.h. dieser wird zur Zeit bearbeitet. Jeder Auftrag ordnet einem Arbeitsgang eine Technologie zu. Verzweigungen innerhalb eines Fertigungsablaufs (z.B. für Nachbearbeitung) lassen sich mit ***Verzweigungsbausteinen*** realisieren.

Der Ablauf der Simulation verläuft in ROSI ereignisorientiert nach der Architektur einer DES. Es existieren mehrere Ereignistypen, die den Transfer und die Bearbeitung von Aufträgen signalisieren. Eine Optimierung des Ablaufplanes, der sich aus der Simulation der Aufträge ergibt, läßt sich durch Veränderung der Prioritäten erreichen. Abbildung 16 zeigt XROSI mit einem Beispielmodell. Auf der rechten Seite werden die einzelnen Maschinen (große Rechtecke) mit den zugehörigen Warteschlangen (darüber, kleine Rechtecke) dargestellt. Die verschiedenen Farben symbolisieren die Zustände der Maschinen. Auf der linken Seite wird der Simulationsfortschritt als Gantt-Diagramm visualisiert.

Durch die oben erläuterten Elemente läßt sich in ROSI eine auftragsgesteuerte Fertigung modellieren. Die Mittel zur Modellierung sind bei weitem nicht so mächtig wie in ARENA oder SIMPLE++, was aber auch nicht das Ziel von ROSI darstellt. Worauf sich ROSI spezialisiert, ist das direkte Einbinden der Auftragssteuerung in die Simulation, indem bereits Strukturen für die Bearbeitung von Aufträgen in der Reihenfolge ihrer technologisch notwendigen Arbeitsschritte bereitgestellt werden.

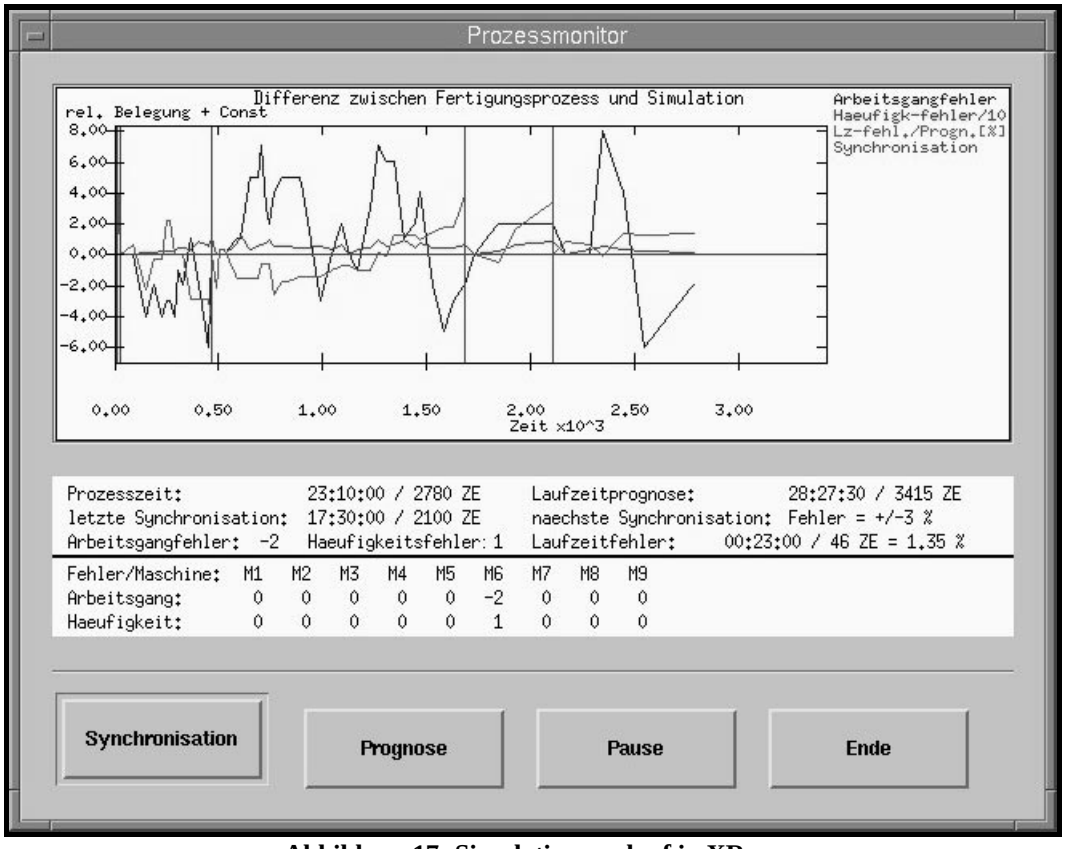


Abbildung 17: Simulationsverlauf in XROSI

Zusätzlich zum einfachen Simulationsablauf bietet ROSI die Möglichkeit, einen durch den Simulationsablauf entstehenden Schedule mit einem vorgegebenen Ausgangsplan zu vergleichen. Dies geschieht zur Laufzeit der Simulation. Bei zu großen Abweichungen von Ausgangsplan und Simulationsablauf kann eine Umplanung ausgelöst werden, die den Ausgangsplan an den Simulationslauf anpaßt. Die Umplanung wird in ROSI ***Synchronisation*** genannt. Dies wird in ROSI zu folgendem Vorgehen verwendet:

- Man bildet ein Simulationsmodell, das das Verhalten des Fertigungsprozesses vom aktuellen Zustand des realen Produktionsablaufes in die Zukunft vorhersagt, und somit einen Ablaufplan für die Zukunft erstellt.
- Der reale Fertigungsprozess bekommt den durch die Simulation erstellten Plan als Vorgabe.

- Diese Vorhersage wird kontinuierlich mit dem realen Ablauf verglichen.

In der Regel führt dies dazu, daß der durch die Simulation erstellte Plan vom realen Ablauf abweicht, da der reale Ablauf Störungen und unvorhergesehenen Ereignissen unterliegt. Bei zu großen Abweichungen wird eine Synchronisation durchgeführt. Abbildung 17 zeigt den Verlauf einer Simulation mit den verschiedenen Synchronisationszeitpunkten, die durch senkrechte Linien gekennzeichnet sind. In Abschnitt 4.6 wurde erläutert, wie die Abweichung gemessen wird. Informationen zur Architektur und Arbeitsweise von ROSI sind in [Sau95] und in der Online-Dokumentation von XROSI zu finden.

5.5. Zusammenfassung

Die vorgestellten Simulationswerkzeuge zeigen eine unterschiedliche Eignung für ihre Verwendung als Grundlage für diese Arbeit. Dabei muß ROSI als akademische Fallstudie betrachtet werden, die zwar einige interessante Ansätze wie die dynamische Bewertung von Belegungsplan und Simulationslauf bietet, aber aufgrund der fehlenden Erweiterbarkeit und Integrationsmöglichkeiten in PPS-Systeme nicht als Grundlage der zu implementierenden Simulationsshell dienen kann.

ARENA zeichnet sich durch eine hohe Performanz und eine große Modellbibliothek aus. Durch die Möglichkeiten der Daten- und Prozeßkommunikation über VisualBasic und Excel steht die grundsätzliche Möglichkeit zur Integration in ein PPS-System zur Verfügung. Allerdings ergeben sich hierbei die Nachteile, daß zusätzlich zur vorhandenen Simulationsumgebung sehr viel Funktionalität neu bzw. doppelt programmiert werden muß und die implementierte Lösung über eine größere Anzahl von Applikationen verstreut ist. Beispielsweise muß sowohl das PPS-System als auch die Simulation ein Modul zur Zeitberechnung anhand von Schichtplänen besitzen. Ebenso müssen Daten aus dem PPS-System gelesen und über die Datenschnittstelle in die Simulation gebracht werden Durch diese Aufteilung sinkt die Performanz der Simulation und macht die Implementation wenig erweiterbar. Ein weiteres praktisches Problem ergibt sich aus den Beschränkungen der Studentenversion von ARENA.

SIMPLE++ stellt das für die gegebene Problemstellung am geeignetsten erscheinende Simulationssystem dar. In SIMPLE++ sind bereits ein Teil der in DYBBS implementierten Fähigkeiten vorhanden, während der Großteil der komplexeren Aufgaben wie die dynamische Bewertung und ereignisgesteuerte Umplanung ebenfalls zu implementieren ist. Durch die Möglichkeiten der objektorientierten Programmierung und den Daten- und Prozeßschnittstellen zu PPS-Systemen ist dies zu realisieren. Als Nachteil ist für die gegebene Aufgabenstellung die Größe des Systems anzusehen. Da SIMPLE++ hauptsächlich von Kunden aus der Großindustrie eingesetzt wird, stehen dort auch genügend Ressourcen zur Ausnutzung dieses Werkzeugs zur Verfügung. SIMPLE++ ist ein großes und sehr komplexes System, das sowohl über eine eigene Programmiersprache als auch über sehr viele Zusatzmodule und Modellbausteine verfügt. Diese Eigenschaften bedingen eine sehr hohe Einarbeitungsdauer in das System und übersteigen den Aufwand einer Diplomarbeit erheblich.

Aus den ausgeführten Gründen wurde im Rahmen dieser Arbeit eine eigene Simulationsshell DYBBS implementiert, die in vielen Bereichen zwar nicht mit kommerziellen Simulationswerkzeugen konkurrieren kann, die aber durch die enge Integration mit dem PPS-Werkzeug WIZARD und der dort vorhandenen Funktionalität die gesetzten Ziele erfüllen kann. Dabei werden viele Ideen aus den vorgestellten Simulationswerkzeugen übernommen und erweitert. Die Architektur von DYBBS ist in Abschnitt 6.3 erläutert.

6. DyBBS

6.1. Einführung in COKE und WIZARD

Die Implementierung von DYBBS erfolgte als zusätzliches Modul des Ressourcenbelegungsplanungstools WIZARD, das seinerseits auf dem konfigurierbaren Expertensystembaukasten COKE aufbaut. Beide Shells sind in Common-LISP implementiert und wurden am Lehrstuhl für Künstliche Intelligenz und Angewandte Informatik an der Universität Würzburg entwickelt. Die ursprünglichen Versionen von COKE und WIZARD wurden unter MCL[13] auf Macintosh-Betriebssystemen erstellt, inzwischen läuft die Weiterentwicklung in einer auf ALLEGRO CL FOR WINDOWS und das Betriebssystem MS Windows 95/NT4.0 portierten Version.

COKE (Coordination with Knowledge-based Expertise) ist eine Shell zur Entwicklung von Expertensystemen für Zuordnungsprobleme, in der verschiedene Problemlöser für die Problemklasse Zuordnung implementiert sind. Dazu gehören die Generiere-und-Optimiere-Strategie, Bestensuche, genetische Algorithmen und Simulated Annealing. Der Schwerpunkt liegt dabei auf der Generiere-und-Optimiere-Strategie, deren Anwendung auf Ressourcenbelegungspläne als „Vorschlagen-und-Vertauschen" in Abschnitt 4.2 vorgestellt wurde. Zur Verwaltung der Objektmengen für Nachfrager und Anbieter bietet COKE ein Objektsystem mit graphischen Eingabemöglichkeiten an. Dieses Objektsystem baut allerdings nicht auf der Basis von CLOS (Common LISP Object System) auf. Im Gegensatz zu CLOS bietet es die Typisierung von Attributen (z.B. Zahlen, Objekt einer Klasse, Text, Termin, Dauer) und eine durchgängige graphische Eingabemöglichkeit für Klassen, Objekte und ihre Attributwerte (s. Abbildung 18). Dafür fehlen die in CLOS vorhandenen Möglichkeiten der Klassenvererbung (die in COKE in anderer Form existiert) und des Methodendispatchings.

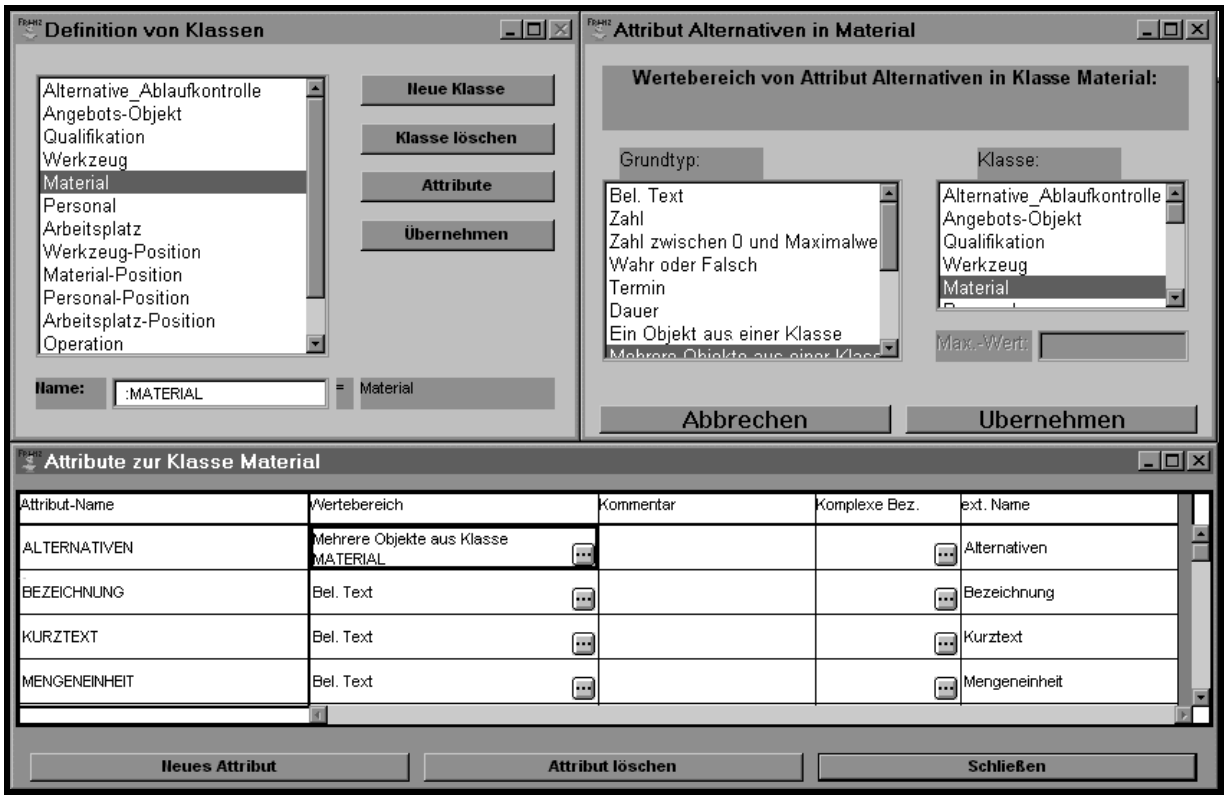


Abbildung 18: Klassendefinition in COKE

Der Ablauf der Zuordnung wird in COKE durch ***Überwachungen*** und ***Vorschläge*** gesteuert. Durch einen ***Dämon*** können Nachrichten an bestimmte Objekte verschickt werden, wenn an einem Objekt ein Attribut verändert wird. Überwachungen (Constraints) enthalten eine Bedingungsfunktion, die darüber

[13] MCL (Macintosh Common LISP) von Digitool, Inc.: http://www.digitool.com/

Auskunft gibt, ob das Constraint verletzt ist, und eine Bewertung über die Schwere der Verletzung. Eine Überwachung wird für ein Attribut einer bestimmten Klasse definiert und bezieht sich bei der Auswertung auf eine Instanz dieser Klasse. Die Bedingungsfunktion wird als LISP-Funktion definiert, wobei in COKE eine Reihe von Prädikatsfunktionen bereitgestellt werden. Wird während der Lösungssuche ein Attribut verändert, für das eine Überwachung aktiviert ist, so wird diese ausgelöst. Ist die Überwachung verletzt, so wird sie in einer Datenstruktur über unerfüllte Überwachungen vermerkt. Der Lösungsalgorithmus kann daraufhin durch einen durch einen Vorschlag ermittelten Korrekturschritt versuchen, die Verletzung des Constraints zu beheben. Die Summe der Bewertungen aller verletzten Überwachungen ergibt eine Bewertung über die Qualität der gefundenen Zuordnung. Aufbauend auf COKE sind der Schulstundenplaner REST und der Ressourcenscheduler WIZARD (s.u.) implementiert worden. Nähere Erläuterungen zu COKE sind in [Poe95] und [Pup90] zu finden. Die Programmierung in COKE wird in [Hes97] beschrieben.

WIZARD (Wissensbasierte Zuordnung in einem Assistenzsystem für die Ressourcendisposition) ist ein auf COKE aufbauender Planer für Ressourcenbelegungspläne. Dazu werden PPS-übliche Massendaten (über Aufträge, Stücklisten, Arbeitspläne, Schichtpläne usw.) und eine Repräsentation des planerischen Wissens des Disponenten benötigt. Da die Massendaten in PPS-Systemen relativ einheitlich modelliert sind, kann WIZARD diese Daten durch eine Datenschnittstelle (z.Zt. über Dateien oder ODBC[14]-Schnittstelle) von einem PPS-System[15] übernehmen. In WIZARD werden Stücklisten und Arbeitspläne zu Ressourcenstücklisten zusammengefasst und können somit simultan behandelt werden. Ebenso werden die vorhandenen Bestände aus den PPS-Daten übernommen. In einer Operation wird zu jedem Arbeitsgang der Bedarf an Ressourcen beschrieben. Die importierten Daten werden in COKE-Objekten abgelegt. Dabei sind die Nachfrageobjekte die ***Operationen***, die den Bedarf an benötigten Arbeitsplätzen, Personal, Materialien und Werkzeugen enthalten. ***Aufträge*** fassen einzelne Operationen zusammen. Im Moment bilden die Operationen in WIZARD einen Baum, d.h. eine Operation kann mehrere Vorgängeroperationen, aber nur einen Nachfolger besitzen. Die Ressourcenklassen ***Arbeitsplätze, Personal, Material*** und ***Werkzeug*** enthalten die Daten über die Angebotsobjekte. Dabei können Arbeitsplätze und Personal zu Gruppen zusammengefasst werden, so daß die Zuordnung auch auf der gröberen Ebene der Arbeitsplatz- und Personalgruppen erfolgen kann. Die Ressourcen Personal und Arbeitsplatz enthalten Verweise auf ***Schichtpläne***, die angeben, wie sich die verfügbare Kapazität einer Ressource über die Zeit verteilt. Aus den PPS-Daten können dann in weiteren Schritten bestimmte Angaben wie z.B. über die voraussichtliche Ressourcenauslastung hergeleitet werden.

Zusätzlich zu den PPS-Daten, die in WIZARD importiert werden, wird eine Repräsentation des planerischen Wissens benötigt. Dabei soll der Wissenserwerb so einfach wie möglich gestaltet werden, so daß das Wissen direkt vom Experten eingegeben werden kann. Auf der elementarsten Ebene stehen die in COKE vorhandenen Restriktionen, Überwachungen, Vorschlags- und Korrekturregeln und die Strategie „Vorschlagen-und-Vertauschen“ (s. Abschnitt 4.2.1), die zur Planerstellung verwendet wird. Der Wissenserwerb auf dieser Ebene besteht hierbei aus der Programmierung und ist Aufgabe der Systementwickler. Der Experte kann aufbauend auf diesen grundlegenden Funktionen eine Auswahl unter den Bausteinen treffen und die Parameter dieser Bausteine verändern. Dadurch kann er die Eigenschaften der Ressourcen (z.B. Zuverlässigkeit von Maschinen) und nicht in den PPS-Daten vorhandenes Wissen (z.B. über besondere Restriktionen) spezifizieren. Konflikte, die bei der Zuordnung nicht aufgelöst werden können, kann der Disponent interaktiv mit dem Planungssystem lösen. Beispielsweise kann er über die Änderung von Endterminen oder die Erhöhung von Kapazitäten (z.B. durch Ansetzen einer Sonderschicht) entscheiden. Diese Entscheidungen können i.a. nicht automatisch getroffen werden, sondern hängen von der Erfahrung des Disponenten und der aktuellen Situation ab. Daher soll das System an dieser Stelle den Experten möglichst gut bei einer manuellen Korrektur unterstützen, indem es die Konsequenzen der Änderungen aufzeigt.

[14] ODBC = Open Database Connectivity: Datenbank-Schnittstelle unter Microsoft-Betriebssystemen

[15] WIZARD ist im Rahmen des INKAD-Projekts (http://www-info6.informatik.uni-wuerzburg.de/~inkad) an das PPS-System VPPS (http://www.infor.de/) gekoppelt worden.

WIZARD bietet durch diese Veränderungsmöglichkeiten zur Planungszeit sehr gute Möglichkeiten zur Any-Time-Planung. Ein wichtiger Bestandteil von WIZARD ist die graphische Präsentation von Ressourcenbelegungsplänen, um die gefundene Zuordnung darzustellen, und dem Disponenten interaktiv Änderungen zu ermöglichen. Zur Plandarstellung können Durchlaufdiagramme, Gantt-Diagramme und Kapazitätsgebirge, auch für mehrere Ressourcen simultan, verwendet werden. Kapazitätsgebirge und Gantt-Diagramme sind in PPS-Systemen übliche Darstellungsformen, während Durchlaufgebirge erstmalig in WIZARD implementiert werden ([Hes95], S. 10). Die drei Darstellungsformen werden in [Kur93] erläutert. Weitere Informationen zu WIZARD, dessen Entwicklung noch nicht abgeschlossen ist, sind in [Hes95] und [Hes97-2] zu finden.

6.2. Begriffsdefinitionen

Gegeben sind anhand der PPS-Daten eine Menge von ***Arbeitsplätzen***, ***Personal***, ***Material***, ***Werkzeug***, ***Aufträge*** und ***Operationen***. Dabei werden wie in PPS-Umgebungen üblich die ersten vier Daten als ***Ressourcen***, die beiden letzten als ***Fertigungsstrukturen*** bezeichnet. Durch das Planungstool WIZARD wird daraus ein Belegungsplan erstellt, der ebenfalls in das Simulationsmodell eingeht.

Zusätzlich zu Ressourcen und Fertigungsstrukturen existieren simulationspezifische Klassen. Diese haben für den Simulationsablauf eine besondere Bedeutung. Diese Elemente (***Simulator***, ***Warteschlange***, ***Welt***, ***Quelle***, ***Senke***) werden als Simulationslemente bezeichnet. Ressourcen, Fertigungsstrukturen und Simulationselemente werden als ***Modellelemente*** (auch: ***Komponenten***, ***Elemente***) zusammengefasst.

Auf der anderen Seite werden durch ***Ereignisse***, ***Zustände***, ***Regeln*** und ***Zustandsgraphen*** das Verhalten der Elemente modelliert. Daher werden diese Strukturen als ***(Simulations-)Wissen*** bezeichnet. Modellelemente und Wissen bilden zusammen die ***Modellbestandteile***. Abbildung 19 zeigt eine Übersicht über die Strukturen in DYBBS. Nicht in der Abbildung enthalten sind allerdings die Zusammenhänge, die zwischen den einzelnen Strukturen vorhanden sind. So besitzt jedes Modellelement einen Zustandsgraph, der wiederum aus Ereignissen, Zuständen und Regeln besteht.

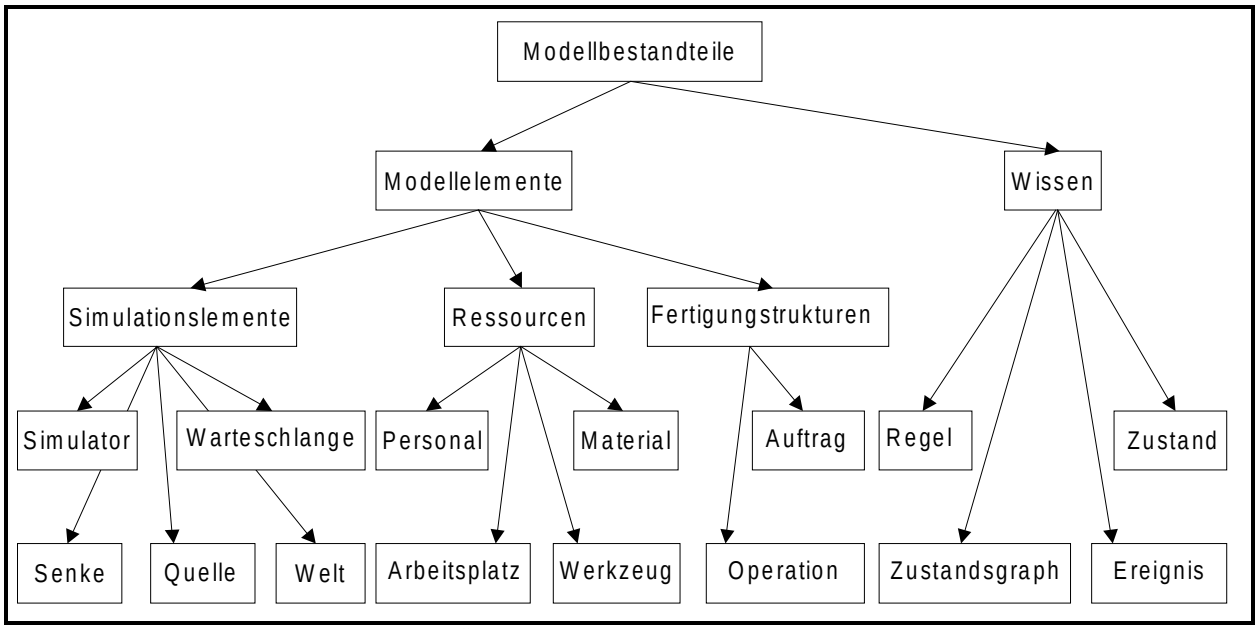


Abbildung 19: Übersicht über die Bezeichnungen in DYBBS

6.3. Architektur und Implementierung

6.3.1. Konzepte

In Abschnitt 5.5 wurde anhand von bestehenden Simulationsshells erläutert, warum im Rahmen dieser Arbeit eine eigene Simulationsshell entwickelt wurde. In diesem Abschnitt werden die Anforderungen, denen das Simulationstool gerecht werden muß, zusammengefasst. [Sch87] richtet an eine in ein Fertigungssystem integrierte Simulation folgende Anforderungen:

- Prozeßschnittstelle zur Online-Übernahme des realen Systemzustandes
- Erreichen eines Echtzeitfaktors (d.h. das Verhältnis zwischen simulierter und realer Prozeßablaufgeschwindigkeit), der erheblich größer als 1 ist.
- Ausreichende Abbildungsgenauigkeit der zugrundeliegenden Prozesse
- Zustandsarchivierung zu definierten Zeitpunkten zur Systeminitialisierung
- Komfortable, möglichst graphische Benutzeroberfläche
- Gezielter Zugriff auf Modellparameter und -zustände

Für wünschenswert hält er die Animation der Abläufe, Bewertungsmöglichkeiten und Planungsunterstützungsfunktionen. In DYBBS sind außer der Animation alle Anforderungen erfüllt. Weiterhin liegt in DYBBS der Schwerpunkt auf einer übersichtlichen Modellierung und einer Konzentration auf den Informationsfluß. Aus diesem Grund kann auf eine Animation, die hauptsächlich den Materialfluß visualisiert, verzichtet werden.

Das Simulationstool soll die Modellierung eines Fertigungssystems auf einer hohen Abstraktionsebene ermöglichen. Im Vordergrund steht dabei der Informationsfluß, nicht der Materialfluß durch das System. Dazu bietet sich eine diskrete ereignisorientierte Simulation an, die das Verhalten einzelner Objekte (z.B. Maschinen) oder Objektgruppen (z.B. Maschinengruppen) durch Ereignisse beschreibt. Das Verhalten wird dabei auf Zustände und Ereignisse der Form „Maschine x beginnt Operation y“, „Maschine x fällt aus“ abstrahiert. Diese Beschreibung ohne Angabe der Details, wie die Operation y auf der Maschine x ausgeführt wird, ergibt ein Simulationsmodell, das dynamisch, zustandsdiskret und zeitkontinuierlich ist. Auf ein solches Modell kann der DES-Algorithmus (s. Abschnitt 2.3) angewendet werden.

Eine weitere Eigenschaft des Produktionsprozesses ist, daß er hochgradig parallel ist. Es erfolgen stets mehrere Produktionsschritte gleichzeitig auf verschiedenen Maschinen, die mehr oder weniger stark in Verbindung stehen. Für eine auftragsgesteuerte Fertigung ergeben sich im Gegensatz zur Fleißfertigung die Verbindungen zwischen den einzelnen Ressourcen erst durch die eingeplanten Aufträge. Das Simulationssystem muß diese Parallelität berücksichtigen, indem eine Modellierung auf der Grundlage der einzelnen Objekte und nicht aufgrund des Gesamtsystems erfolgt, und muß demzufolge auch Möglichkeiten zur Synchronisation der einzelnen Prozesse bieten. In Abschnitt 3.4.3 wurde mit den Kontrollflußgraphen ein Modellierungsansatz vorgestellt, der speziell auf eine Modellierung paralleler Prozesse abgestimmt ist.

Da innerhalb eines Fertigungssystems (und daher auch in den Daten eines PPS-Systems) eine große Anzahl von Objekten vorliegt, müssen geeignete Methoden zur Gruppierung und Hierarchisierung verwendet werden. Eine Gruppierungsmöglichkeit bietet sich in den verschiedenen Klassen der PPS-Daten (Aufträge, Operationen, Arbeitsplätze, Personal, Material, Werkzeug) an, eine weitere ist in der Zusammenfassung von Arbeitsplätzen zu Arbeitsplatzgruppen usw. gegeben. Aufgrund dieser Möglichkeiten sollte das Simulationssystem eine Modellierung mit einer zu den PPS-Daten ähnlichen Strukturierung bieten. Dies ist auch aus dem Grund, daß die Anwender mit dieser Strukturierung ihrer Daten bereits vertraut sind, sinnvoll. Daher erfolgt die Modellierung in einem hierarchischen Modell, das auf der obersten Ebene eine Übersicht über die vorhandenen Aufträge, Arbeitsplatz- und Personalgruppen zeigt. Jede Arbeitsplatz- und Personalgruppe wird auf der nächsten Stufe in ihre Einzelressourcen gegliedert,

für jeden Auftrag wird der Baum der zu diesem Auftrag gehörenden Operationen dargestellt. Auf der untersten Ebene erfolgt dann die Modellierung eines einzelnen Objekts.

In Abschnitt 4.2.1 wurde erläutert, daß die einzelnen Objekte der PPS-Daten eine große Anzahl von Attributen aufweisen. Die meisten dieser Attribute sind jedoch für den Systemzustand eines Objektes während der Simulation im Sinne von Abschnitt 2.1 nicht von Bedeutung. Daher muß der für die Simulation wichtige Zustand eines Objektes durch eine geeignet knappe und aussagekräftige Zustandsmodellierung erfolgen. Die in Abschnitt 3.3 beschriebene Modellierung über Petri-Netze bietet eine solche Zustandsbeschreibung. Während in Petri-Netzen das Verhalten durch das Feuern von Übergängen anhand von Marken formal festgelegt ist, bieten Ereignisgraphen an dieser Stelle durch die Angabe von Regeln mehr Modellierungsfreiheit. Allerdings sind in Ereignisgraphen die Regeln durch Code-Fragmente spezifiziert, was sie wenig übersichtlich und intuitiv macht.

Da das Simulationssystem zur Unterstützung der Belegungsplanung geplant ist, soll es im Gegensatz zu einmaligen Simulationsstudien ständig in Verwendung sein. Daher muß eine gute Integration zu den PPS-Daten gegeben und die Simulation hinreichend effizient implementiert sein, um während der normalen Disposition eingesetzt zu werden.

Für die Modellierung in DyBBS wurde daher eine Modellierungsmethode entwickelt, die aus den vorhandenen Methoden DEVS, Ereignisgraphen und Petri-Netze Elemente kombiniert, um für die Modellierung eines Fertigungsystems eine angemessene Repräsentation zu erhalten. Die hierarchische, objektorientierte Gliederung aus DEVS wurde übernommen, wobei sich die Hierarchiebildung nach den in den PPS-Daten abgelegten Datenstrukturen richtet. Für die Modellierung der einzelnen Elemente wurden die Ereignisgraphen um das in Petri-Netzen vorhandene Zustandskonzept erweitert. Durch diese Kombination erhält man eine Methode, die in graphisch übersichtlicher Weise sowohl Zustände und Ereignisse als auch die Regeln, die die Zustandsübergänge steuern, explizit darstellen, ohne daß formale Beschränkungen wie in Petri-Netzen die Modellierung erschweren. Für die Verhaltenssteuerung durch Regeln wurde ein mächtiges und graphisch unterstütztes Regelkonzept entwickelt, das die Angabe der Regeln im Vergleich zu Ereignisgraphen erheblich vereinfacht.

6.3.2. Realisierung

DYBBS besteht aus drei großen Modulen, die aufeinander aufbauen. Die Basis bildet das Statistikmodul, das grundlegende statistische Funktionen wie Verteilungsfunktionen und Stichproben bereitstellt. Im Simulationskern befindet sich die Implementation des DES-Algorithmus und die graphische Benutzeroberfläche zur Modellierung. Diese beiden Bereiche verwenden das in COKE vorhandene Objektsystem und dessen Möglichkeiten zur graphischen Objekteingabe. Auf dieser Stufe steht eine Simulationsshell für DES zur Verfügung, die für allgemeine Simulationsstudien verwendet werden kann.

Die oberste Schicht in DYBBS bildet die Integration in WIZARD. Diese ermöglicht die teilweise automatische Erstellung von Modellen aus PPS-Daten sowie die Verwendung von Belegungsplänen. Ebenso gehören in diese Schicht die Bewertung von Belegungsplänen, die durch die Simulation erstellt wurden, der Vergleich zwischen Simulationsablauf und Belegungsplan und die Umplanung während eines Simulationslaufes. Abbildung 20 zeigt die Architektur von DYBBS und die Beziehungen der einzelnen Module zueinander. In den folgenden Abschnitten werden die wichtigsten Aufgaben, Klassen und Funktionalität der einzelnen Module beschrieben, in Anhang D befindet sich eine Übersicht über die Schnittstellen der einzelnen Module.

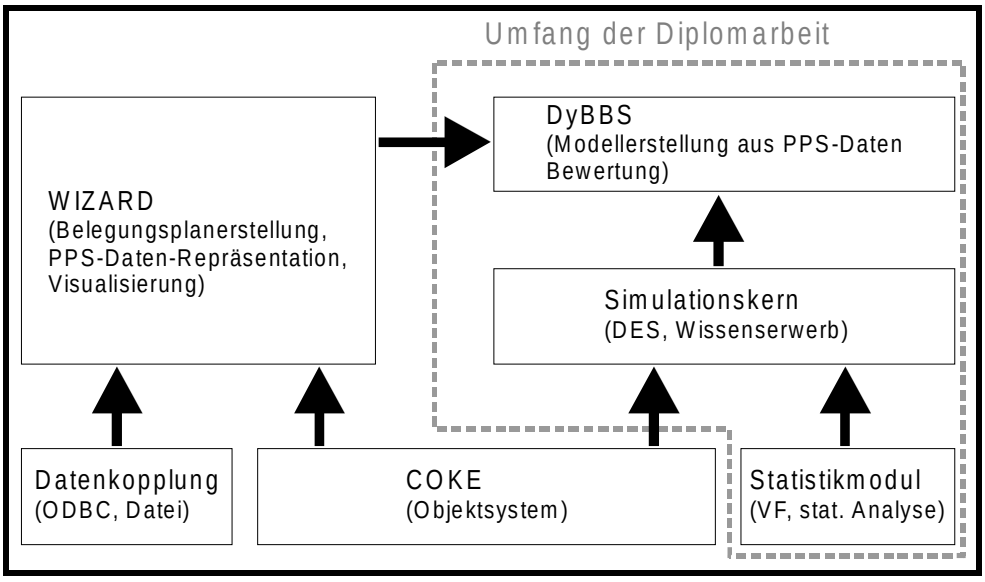


Abbildung 20: Systemarchitektur in DYBBS

6.3.3. Statistikmodul

Das Statistikmodul enthält grundlegende Klassen und Funktionen zur Erzeugung von Zufallszahlen und zur statistischen Analyse der Simulation. Dieses Modul hängt nicht von den anderen Systemen ab und kann für andere Zwecke wiederverwendet werden. Seine Aufgaben können in drei Gruppen eingeteilt werden:

- Erzeugung von Zufallszahlen nach vorgegebenen Verteilungen
- Bereitstellung von Stichproben
- statistische Analyse der Stichproben

Die Strategien zur Erzeugung von Zufallszahlen und die implementierten Verteilungsfunktionen sind in Anhang C erläutert. Insgesamt wurden 12 Verteilungsfunktionen (z.B. Normal-, Exponential- oder Dreiecksverteilung) implementiert, deren Erzeugungsalgorithmen in [Law91] und [Hoo89] beschrieben sind. Dort finden sich auch Algorithmen für weitere Verteilungen, die bei Bedarf ebenfalls implementiert werden können.

Die zweite Aufgabe des Statistikmoduls ist die Verwaltung von Stichproben. Dabei werden die (zufallsabhängigen) Daten, die während der Simulation anfallen, gesammelt und können zur Auswertung der Simulationsergebnisse analysiert werden. Diese Daten betreffen z.B. die Verweildauer einzelner Elemente in ihren Zuständen oder die Länge von Warteschlangen. Für die Analyse wurde nur ein Minimalumfang zur statistischen Analyse implementiert. Die Statistikfunktionen beschränken sich auf die Berechnung von Mittelwert, Minimum, Maximum, Varianz, Standardabweichung und Korrelationskoeffizient. Es besteht allerdings die Möglichkeit, die Simulationsergebnisse zu exportieren und mit anderen Programmen detaillierter auszuwerten.

6.3.4. Simulationskern

Der Simulationskern von DYBBS läßt sich in zwei Untermodule gliedern. Das DES-Modul implementiert den DES-Algorithmus und dient der Steuerung des Simulationsablaufs. Im Modul Wissenserwerb ist die Modellerstellung implementiert. Der Wissenserwerb liefert ein Simulationsmodell, das im DES-Modul simuliert wird. In diesem Sinne baut der Wissenserwerb auf das DES-Modul auf. Im Prinzip wäre es möglich, den implementierten Wissenserwerb (zumindest teilweise) durch eine andere Modellierungsmethode (s. Kapitel 3) zu ersetzen.

DES-Modul

Dieses Modul implementiert den DES-Algorithmus aus Abschnitt 2.3. Somit sind die Hauptaufgaben des DES-Moduls:

- Initialisierung der Simulation
- Erzeugen, Verwalten und Ausführen von Ereignissen
- Verändern des Zustands über die mit den Ereignissen verknüpften Regeln
- Erzeugen und Verwalten der statistischen Elemente (Stichproben, Verteilungsfunktionen) des Statistikmoduls
- Beenden der Simulation und Aufbereitung der statistischen Daten

Die Implementierung des DES-Modul erfolgt hauptsächlich in den Klassen Ereignis, Simulator und den grundlegenden Methoden der Elementklassen. Der Ablauf der Simulation stellt den zeitaufwendigsten Teilbereich der Simulation dar. Daher muß die Implementierung effizient (v.a. in Bezug auf die Laufzeit) sein, weshalb während der Simulation die Kommunikation mit dem Benutzer sowie die Darstellung des Simulationsverlaufs auf ein Minimum beschränkt wird. Der Benutzer kann erst nach dem Anhalten der Simulation interaktiv auf den Simulationsverlauf einwirken.

Allerdings bleibt festzustellen, daß die Simulationsdauer in DYBBS um einiges[16] höher ist als in den in Abschnitt 5.2 und 5.3 vorgestellten kommerziellen Simulationsumgebungen. Dies ist u.a. mit der gewählten Implementationsumgebung (LISP, ALLEGRO CL 3.0.2[17]) zu erklären.

Ereignis

Jedes Ereignis ist ein eigenes Objekt, das aufgrund seines Eintretens an einem Umgebungselement eine Zustandsänderung eines Elements oder eine andere Aktion ausführen kann. Ereignisse werden von einem Erzeuger zu einem Zeitpunkt angeregt und zu einem Ziel geschickt, das die Behandlung des Ereignisses übernimmt. Die Auswahl des nächsten abzuarbeitenden Ereignisses übernimmt die Agenda des Simulators. Folgende Attribute werden für Ereignisse benötigt:

- `typ:` Art des Ereignisses. Die verschiedenen Ereignistypen können im Wissenserwerb frei definiert werden.
- `zeitpunkt:` Zeitpunkt des Ereignisses.
- `prioritaet:` bei gleichem Zeitpunkt mehrerer Ereignisse entscheidet die Priorität, welches zuerst behandelt wird.
- `inhalt:` Jedem Ereignis kann als Inhalt (im Sinne einer Botschaft) ein beliebiges Datum mitgegeben werden.
- `ausloeser:` ein Umgebungselement, welches das Ereignis auslöst.
- `ziel:` ein oder mehrere Umgebungselemente, welche das Ereignis behandeln. Quelle und Ziel lassen sich dynamisch aufgrund der Regeln bestimmen. Insbesondere können Ziel und Auslöser identisch sein.

Das Erzeugen von Ereignissen erfolgt durch spezielle Auswirkungen (s. Regeln, S. 75). Ereignisse, die aufgerufen, aber nicht ausgeführt werden, weil die Vorbedingung nicht erfüllt ist, können zwei Verhalten zeigen:

- Das Ereignis wird nicht ausgeführt und verworfen.

[16] Beim Vergleich zwischen ARENA und DYBBS für das einfache M/M/1-Warteschlangenmodell (s. Abschnitt 7.1) war die Laufzeit ca. 50 mal größer. Dieser Abstand würde sich allerdings für komplexe Modelle mit vielen Aufrufen der PPS-Funktionen reduzieren.

[17] In bestimmten numerischen Benchmarks ist die Linux-Version von Allegro CL auf dem gleichen Rechner um den Faktor 5-10 schneller.

- Das Ereignis wird nicht weggeworfen. Es wird in einer besonderen Datenstruktur des Simulators gespeichert. Dazu wird vermerkt, auf welche anderen Ereignisse hin das gespeicherte Ereignis wieder aktiviert werden soll. Bei Eintreten dieses Ereignisses wird dann wiederum die Vorbedingung geprüft.

  Beispiel: eine Operation kann nur an einem Arbeitsplatz X ausgeführt werden, wenn alle Vorgängeroperationen beendet worden sind. Somit wartet das Ereignis „Bearbeitungsbeginn(X)“ auf ein Ereignis vom Typ „Bearbeitungsende“. Dabei wird jedes Ereignis von Typ „Bearbeitungsende“ dazu benutzt, das wartende Ereignis wieder zu aktivieren. Ob die beendete Operation auch tatsächlich die Operation ist, auf die gewartet wurde, muß in diesem Fall gesondert überprüft werden.

Simulator

Der eigentliche Simulator ist ebenfalls ein Simulationselement. Im Gegensatz zu den anderen Simulationselementen muß es immer genau ein Objekt dieser Klasse im Simulationsmodell geben, da dieses den Ablauf der Simulation steuert. Damit kann das Verhalten des Simulators auf die gleiche Weise modelliert werden wie die anderen Elemente. Insbesondere können dem Simulator selbst Ereignisse geschickt werden, die dieser behandeln kann. So können auf einfache Weise Start, Stop, Unterbrechen und Animation der Simulation umgesetzt werden. Bei Bedarf kann auch das Verhalten des Simulators im Wissenserwerb erweitert werden. Der Simulator benötigt folgende spezielle Attribute:

- `agenda:` alle noch abzuarbeitenden Ereignisse. Die Agenda ist als Warteschlange implementiert.
- `starttermin:` Startzeitpunkt der Simulation
- `stoptermin:` Endzeitpunkt der Simulation
- `simulationselemente:` alle in der Simulation verwendeten Elemente
- `aktueller-simulationspunkt:` Wert der „globalen Uhr“
- `wartende-ereignisse:` die Menge der Ereignisse, die auf die Reaktivierung wartet.

Wissenserwerb

In diesem Abschnitt werden die Klassen beschrieben, die zur Modellierung in DYBBS bereitgestellt werden. Dabei werden zum einen die PPS-üblichen Datenstrukturen erweitert und zum anderen neue Datenstrukturen hinzugefügt, die zur Modellerstellung benötigt werden. Der Wissenserwerb erfolgt in DYBBS graphisch in Interaktion mit dem Benutzer, um eine möglichst einfache und intuitive Modellierung zu ermöglichen. Ein ausführliches Handbuch zur Bedienung von DYBBS befindet sich in Anhang B und ist als kontextsensitive Hilfe in das System eingebunden.

Da in einem Modell einer Fertigung relativ viele Objekte mit bestimmten, oft ähnlichen Eigenschaften vorkommen, erfolgt die Modellierung hierarchisch. Die Modellierung erfolgt für jedes Objekt bzw. für eine Klasse von Objekten, was eine ausreichende Übersichtlichkeit gewährleistet. Ausgehend von den in Kapitel 3 vorgestellten Modellierungsmethoden und der in der Zusammenfassung dieser Methoden aufgestellten Anforderungen an die Modellierung (s. Abschnitt 3.6 und 6.3) wurde eine eigene Modellierungsmethode entwickelt, die aus den vorgestellten Modellierungsmethoden DEVS, Event Graphs und Petri-Netzen Elemente übernimmt und sie miteinander verknüpft.

Die hierarchische Modellierungsmethode aus DEVS wird insoweit übernommen, als die Modellierung für jedes Objekt des Simulationsmodells (z.B. einen Arbeitsplatz) getrennt erfolgt. Die Beziehung der einzelnen Objekte wird im Nachfolgergraph (s. Anhang B.1) dargestellt, der die logische Reihenfolge der zu bearbeitenden Operationen und die Aufteilung der PPS-Ressourcen in Gruppen darstellt. Für den Benutzer, der mit der zugrundeliegenden Fertigungsstruktur vertraut ist, ergibt sich somit eine intuitive Gliederung des Modells. Die Erstellung der Hierarchie erfolgt automatisch aus den PPS-Daten. Auf der obersten Hierarchieebene des Modells werden die geladenen Aufträge, Arbeitsplatz- und Perso-

nalgruppen und einzelne simulationspezifische Elemente dargestellt. Für jede Arbeitsplatz- und Personalgruppe besteht die nächste Hierarchiestufe aus den Einzelressourcen der Gruppe. Ein Auftrag wird auf der nächsten Ebene in den Baum der jeweiligen Operationen gegliedert. Auf der Modellierungsebene der einzelnen Objekte kann das Verhalten sowohl für Gruppen- als auch Einzelressourcen definiert werden.

Die verschiedenen Ressourcenklassen und Fertigungsstrukturen können aus den PPS-Daten übernommen werden. Für zusätzliche Funktionalität stehen die simulationsspezifischen Elemente zur Verfügung. Beispielsweise dient eine Quelle dazu, um Operationen innerhalb der Fertigung freizugeben, ein Welt-Objekt kann zur Modellierung von Ereignissen, die ansonsten nicht im System erfaßt wären, verwendet werden. Jedes Element besitzt aufgrund seiner Klasse bestimmte Eigenschaften, die in besonderen Dialogen festgelegt werden können (s. Anhang B.3). Zusätzlich dazu können für jedes Element vom Benutzer eigene Attribute definiert werden, so daß in Verbindung mit der Parametrisierung von Regeln (s.u.) eine Wiederverwendung von Teilmodellen möglich ist.

Eine besondere Rolle spielen Warteschlangen, von denen jeweils eine mit jeder Ressource verknüpft ist. Die Ressourcen werden als statische Elemente, die Anforderungen erhalten und bedienen müssen, angesehen, während die Fertigungsstrukturen die dynamischen Elemente darstellen, die als Anforderungen von den Ressourcen bedient werden müssen. Liegen an einer Ressource mehr Anforderungen vor als sofort bedient werden können, so müssen einige Anforderungen warten. Welche Anforderungen bedient werden und welche warten müssen, entscheidet die Warteschlangendisziplin der entsprechenden Warteschlange.

Zustandsgraph

Die einzelnen Elemente werden durch eine Modellierungsmethode dargestellt, für die die Bezeichnung Zustandsgraph gewählt wurde. Dieser stellt eine Erweiterung der Ereignisgraphen um Zustände dar, wobei ähnlich wie in Petri-Netzen Ereignisse und Zustände abwechselnd einen Graph aufbauen. Ein Objekt befindet sich in einem Zustand, den es aufgrund des Eintretens eines bestimmten Ereignisses verlassen kann und in einen neuen Zustand übergehen kann. Weitere Nebenbedingungen für den Zustandsübergang und die Veränderungen, die ein Zustandsübergang auslöst, sind durch Regeln spezifiziert. Anders als in Ereignisgraphen ist der Zustand eines Objekts somit explizit dargestellt. Im Gegensatz zu Petri-Netzen existieren im Zustandsgraph keine Marken, die den Übergang von einem Zustand zum nächsten veranlassen. Der Übergang erfolgt durch das Eintreffen von Ereignissen.

In DYBBS wird jedem Element des Simulationsmodells ein Zustandsgraph zugewiesen. Somit haben sowohl die PPS-Ressourcen (Arbeitsplätze, Personal, Material, Werkzeug), die Fertigungsstrukturen (Aufträge, Operationen) als auch die simulationsspezifischen Elemente (Simulator, Warteschlange, Quelle, Senke, Bewerter) einen eigenen Zustandsgraph und können auf die gleiche Weise modelliert werden. Da oft mehrere Objekte sich in ihrem Verhalten nicht unterscheiden (z.B. alle Maschinen eines bestimmten Typs), kann die Modellierung dieser Elemente gemeinsam erfolgen. Diese Elemente teilen sich dann auch alle Datenstrukturen, die während des Simulationsablaufes statisch sind, wodurch erheblich Speicherplatz eingespart wird. Zur Beschreibung eines Zustandsgraphen werden folgende Attribute benötigt:

- `zustaende:` die Menge der möglichen Zustände
- `startzustand:` der Zustand, in der sich das Element zu Beginn befindet
- `uebergangsrelation:` eine Abbildung, die zu den Argumenten Zustand *z* und Ereignistyp *e* die Menge der möglichen Folgezustände *Z'* liefert. Für jedes Tupel *(z,e,z')*, $z' \in Z'$ kann eine Menge von Regeln angeben werden, die mit diesem Übergang verbunden ist. Diese Menge kann auch leer sein.

- `uebergangswahrscheinlichkeit:` Wahrscheinlichkeit, ob ein Ereignis eintritt. Diese ist unabhängig vom Zeitpunkt des Ereignisses und dient zur Vereinfachung der Modellierung.
- `arbeitszustaende:` eine Teilmenge der Zustände, in der das Element „arbeitet“. Diese Teilmenge ist vom Modellierer festzulegen und wird für die Berechnung der Auslastung benötigt.
- `ausfallzustaende:` Eine ebenfalls vom Modellierer anzugebende Teilmenge von Zuständen, in denen ein Element nicht arbeiten kann, selbst wenn Anforderungen an dieses Element vorliegen (Störung). Je nach Modell und Sichtweise des Modellierers können Arbeits- und Ausfallzustände keinen, einen oder mehrere Zustände umfassen. Beispielsweise kann der Zustand „Rüsten“, der das Rüsten einer Maschine beschreibt, als Arbeitszustand gezählt werden.
- `statistiken:` über die Verweildauer in den einzelnen Zuständen.

Die Semantik, die einem Zustandsgraph zugrunde liegt, ist die folgende:

1. Zu Beginn der Simulation befindet sich das Element in seinem Startzustand.
2. Trifft am Element, das sich im Zustand *z* befindet, ein Ereignis vom Typ *e* ein, so wird geprüft, ob Folgezustände *Z'* für das Paar *(z,e)* existieren. Gibt es keine möglichen Folgezustände, so wird das Ereignis ignoriert.
3. Existieren Folgezustände *Z'*, so werden der Reihe nach alle möglichen Folgezustände überprüft. Zu einem Tupel $(z,e,z'),z' \in Z'$ wird die Menge aller mit diesem Übergang verknüpften Regeln ermittelt. Die Regeln werden wiederum der Reihe nach getestet. Die erste Regel, die feuert, löst den Übergang des Elements in den Zustand *z'* aus. Danach wird keine weitere Regel und kein anderer Folgezustand *z''* mehr verfolgt. Existiert keine Regel zu diesem Übergang, wird sofort der Folgezustand angenommen.
4. Schritt 2 und 3 werden solange wiederholt, bis die Simulation endet.

In Schritt 3 ist weder die Reihenfolge der Folgezustände noch die der Regeln festgelegt. Will der Modellierer ein deterministisches Verhalten bei den Übergängen erreichen, müssen die Regeln so formuliert werden, daß immer nur eine Regel feuern kann. In vielen Fällen reicht es allerdings aus, zu jedem möglichen Übergang eine Regel anzugeben. Offensichtlich ist der Zustand eines Element zu jedem Zeitpunkt bekannt, allerdings nicht der nächste Zustand, den es annimmt. Dieser hängt von dem ankommenden Ereignis und den Regeln, die sich daraus ergeben, ab. Die möglichen Zustandsübergänge definieren die Kanten im Zustandsgraph, dessen Knoten sich aus den Zuständen und Ereignissen zusammensetzen.

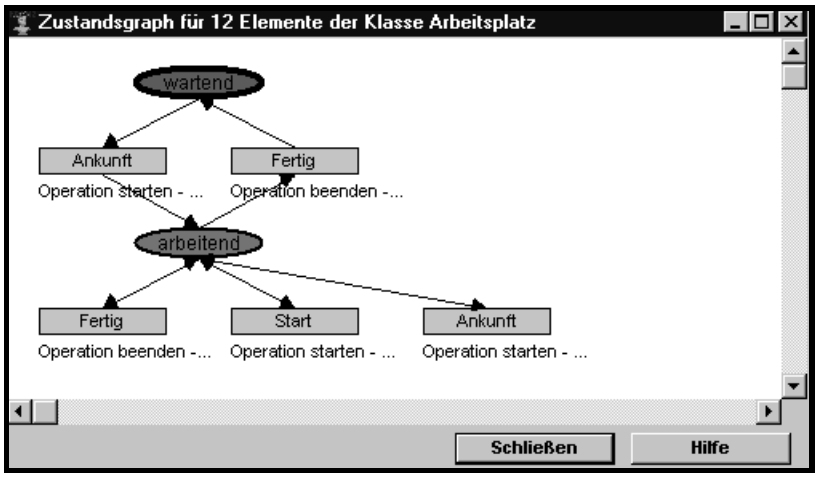


Abbildung 21: Zustandsgraph

Abbildung 21 zeigt ein Beispiel für einen Zustandsgraph mit zwei Zuständen (Ovale) und fünf Ereignissen (Rechtecke). Insgesamt werden in diesem Beispiel drei verschiedene Regeln (Beschriftung

unterhalb der Ereignisse) verwendet. Im Benutzerhandbuch (s. Anhang B.2) wird die Modellerstellung in DYBBS durch Zustandsgraphen detaillierter beschrieben.

Regeln

Regeln dienen zur Beschreibung der Folgen, die Ereignisse an Elementen auslösen. Jede Regel ist einem oder mehreren Ereignistypen zugeordnet. Eine Regel besitzt eine Vorbedingung, deren Erfüllung beim Eintreten des Ereignisses geprüft wird. Wenn diese erfüllt ist, werden die Auswirkungen der Regel ausgeführt. Die Vorbedingung kann in mehrere Einzelbedingungen zerfallen, die über logische Verknüpfungen miteinander verbunden sind. Auswirkungen können eine Zustandsänderung, eine Ereignisgenerierung oder eine allgemeine LISP-Funktion betreffen. Somit haben Regeln folgende Attribute:

- `bezeichnung` und `beschreibung:` Name und Erklärung
- `gehoert-zu:` eine oder mehrere Ereignistypen
- `vorbedingung:` eine logische Aussage, die einzelne Vorbedingungen verknüpft.
- `auswirkungen:` Eine geordnete Menge von Aktionen, die beim Feuern der Regel ausgeführt werden.
- `parameter:` Regeln können in DYBBS parametrisiert werden, was analog zur Parameterübergabe an Prozeduren in einem Programm verläuft. Für Regeln, Auswirkungen und Vorbedingungen können Parameter definiert werden, die analog zu Attributen in COKE-Klassen typisiert sind. Die Parameter einer Regel können bei der Verwendung der Regel im Zustandsgraph mit festen Werten oder mit Modellparametern belegt werden, in Auswirkungen und Vorbedingung einer Regel können zusätzlich auch die Parameter der Regel weitergereicht werden. Auf diese Weise können Regeln, Vorbedingungen und Auswirkungen einfach wiederverwendet werden.

Die Erstellung und Verwendung von Regeln, Vorbedingungen und Auswirkungen über vom Benutzer auszufüllende Formulare ist genauer im Benutzerhandbuch (s. Anhang B.8-B.11) beschrieben. Dort ist auch in Abbildung 33 (S. 109) ein Beispiel einer Regel abgebildet.

Vorbedingungen

Vorbedingungen bestehen aus einzelnen Prädikaten, die über eine logische Formel verknüpft sind. Prädikate sind im wesentlichen Testfunktionen, die entweder wahr oder falsch zurückgeben. Die Verknüpfung der Prädikate kann mithilfe der Makros der COKE-Kommandosprache erfolgen, womit sowohl aussagen- als auch prädikatenlogische Aussagen möglich sind. Für Vorbedingungen werden folgende Attribute benötigt:

- `bezeichnung` und `beschreibung:` Name und Erklärung
- `gehoert-zu:` eine Elementklasse
- `praedikat:` die eigentliche Testfunktion
- `parameter:` Vorbedingungen können wie Regeln parametrisiert werden.

Ein Beispiel für eine Vorbedingung ist wiederum im Benutzerhandbuch in Abbildung 35 (S. 111) dargestellt.

Auswirkungen

Auswirkungen beschreiben die Veränderungen, die bei einem Zustandsübergang vorgenommen werden. In DYBBS wurden insgesamt vier Arten von Auswirkungen implementiert (s. Anhang B.11), die sich durch ihre Eingabeformulare und Spezialisierung voneinander unterscheiden. Insbesondere erfolgt die Generierung von neuen Ereignissen durch spezielle Auswirkungen. Allgemein werden für Auswirkungen die folgende Attribute benötigt:

- `bezeichnung` und `beschreibung:` Name und Erklärung
- `typ:` einer der speziellen Auswirkungstypen: Ereignisgenerierung, Attributsveränderung, Umplanung, allgemeine LISP-Funktion
- `gehoert-zu:` eine Elementklasse
- `aktionen:` die eigentlichen Auswirkungsfunktionen (diese sind z.T. für den Benutzer durch spezielle Formulare verborgen)
- `parameter:` analog zu Regeln und Vorbedingungen

Abbildung 36 und Abbildung 37 (S. 113f) zeigen zwei Beispiele für Auswirkungen. Durch Zustandsgraph und Regeln stehen dem Benutzer zwei sehr allgemeine Konzepte zur Modellierung zur Verfügung, die zum einen eine effiziente Umsetzung auf den DES-Algorithmus und zum anderen eine ausreichende Mächtigkeit und Übersichtlichkeit zur Modellerstellung ermöglichen.

6.3.5. Integration von DyBBS in WIZARD

Während der Simulationskern das von COKE implementierte Objektsystem verwendet, wird in der darüberliegenden Integrationsschicht die Funktionalität von WIZARD in DYBBS eingebunden. Gerade durch die Integration dieser PPS-bezogenen Funktionalität in eine Simulationsshell stellt DYBBS eine Besonderheit dar. Die Integration findet in drei Teilbereichen statt:

- bei der Modellerstellung
- beim Simulationsablauf
- bei der Analyse der Simulationsergebnisse

Im folgenden werden die wichtigsten Aspekte der Integration in diesen Teilbereichen detaillierter ausgeführt.

Modellerstellung

Die Modellerstellung anhand von PPS-Daten ist in DYBBS durch die Kopplungsfunktionen von WIZARD möglich. Dabei werden die PPS-Daten über Ressourcen und Fertigungsstrukturen über eine Datenbankschnittstelle oder Dateiimport in WIZARD in entsprechende Datenstrukturen eingelesen. Diese Datenstrukturen (z.B. einzelne Arbeitsplätze) werden in DYBBS zur Modellierung verwendet und dazu um einige simulationsspezifische Attribute erweitert. Diese betreffen u.a. den Zustandsgraph, der mit dem Element verbunden ist. Die Zusammenhänge zwischen den PPS-Daten, z.B. die Einteilung der Arbeitsplätze in Gruppen- und Einzelarbeitsplätze, werden graphisch in einer hierarchischen Modellübersicht dargestellt. Durch den Import von PPS-Daten ist es möglich, zu jedem Zeitpunkt eine Simulation anhand der aktuellen Betriebssituation zu erstellen. Durch die Trennung der Modellierung in Komponenten und deren einzelne Modellierung durch Zustandsgraphen ist es möglich, neue oder andere PPS-Daten zu koppeln, diese mit bereits erstellten Zustandsgraphen zu verbinden und so ohne Modellierungsaufwand ein ablauffähiges Simulationsmodell zu erhalten. Die Integration von DYBBS in WIZARD erlaubt es somit, ein erstelltes Simulationsmodell mit minimalem Aufwand auf veränderte oder völlig verschiedene Daten anzusetzen.

Simulationsablauf

Während der Simulation werden viele Ereignisse generiert, deren Ereigniszeitpunkte von den PPS-Daten abhängig sind. Die Ereigniszeitpunkte hängen u.a. von den Schichtplänen ab, die mit den Ressourcen verbunden sind. Beispielsweise löst das Ereignis „Operation *X* wird auf Arbeitsplatz *Y* zum Zeitpunkt *Z* gestartet“ ein Folgeereignis *Z'* aus, welches das Ende der Bearbeitung signalisiert. Dessen Zeitpunkt hängt von der Dauer der Operation *X* und dem Schichtplan des Arbeitsplatzes *Y* ab. Wenn *X* die Dauer *4h* besitzt, so berechnet sich der Zeitpunkt von *Z'* aus *Z + 4h* relativ zum Schichtplan von *Y*, was bedeuten kann, das *Z'* sehr viel später als *Z + 4h* liegt, da die Operation von einer Pause unterbrochen

wurde. In DYBBS werden diese Berechnungen automatisch durchgeführt, wobei die entsprechenden WIZARD-Schnittstellen zur Berechnung von schichtplanabhängigen Dauern und Terminen verwendet werden. Der Benutzer muß bei der Modellierung nur angeben, daß der Ereigniszeitpunkt schichtplanabhängig ist. Die Ergebnisse der Simulation sind somit immer reale Daten, d.h. die durch die in der Simulation entstehenden Dauern und Termine von Operationen sind ohne weitere Umrechnung zu lesen. In den meisten Simulationswerkzeuge ist die Verwendung von Schichtplänen nur unter zusätzlichem Modellierungsaufwand möglich.

In DYBBS ist es weiterhin möglich, während des Simulationslaufs eine Bewertung der entstehenden Belegung und eine Umplanung durchzuführen. Diese dynamischen Eingriffsmöglichkeiten sind ebenfalls in anderen Simulationstools üblicherweise nicht vorhanden, sondern müssen zusätzlich programmiert werden. Die Bewertung des Simulationslaufs erfolgt durch das simulationsspezifische Element Bewerter. Dieses ermittelt aufgrund eines Maßes eine Bewertung für den aktuellen Zustand der Simulation. Vorschläge für verschiedene Bewertungskriterien zur Beurteilung des Simulationszustandes wurden in Abschnitt 4.6 vorgestellt. Dabei können jedoch ohne Probleme neue Kriterien hinzugefügt werden. Für die Klasse Bewerter werden zusätzlich zu den allgemeinen Attributen für Elemente folgende Attribute benötigt:

- `letzte-Umplanung:` Termin, an dem das letzte Mal der Plan geändert wurde.
- `mindest-Dauer:` wie lange darf keine Umplanung durchgeführt werden.
- `maximal-Dauer:` wann muß spätestens eine Umplanung durchgeführt werden.
- `bewertungsfunktion:` eine Funktion, die nach den vorgestellten Prinzipien einen numerischen Wert zurückliefert.
- `schranke:` eine (untere und) obere Schranke, innerhalb dessen die Bewertungsfunktion bleiben muß.

Der Benutzer kann im Simulationsmodell durch Ereignisse eine Bewertung auslösen. Dies kann beispielsweise bei jeder Störung einer Maschine geprüft werden.

Wenn die Bewertung eine gewisse Schranke überschreitet, so kann in DYBBS eine Umplanung durchgeführt werden. Dabei werden alle noch nicht abgearbeiteten Operationen nach einer bestimmten Planungsstrategie neu eingeplant. Diese veränderte Planung wird daraufhin zur Grundlage der Simulation gemacht, d.h. für den weiteren Verlauf der Simulation wird mit den umgeplanten Operationen weitergearbeitet. So kann z.B. eine Überlastung eines Arbeitsplatzes dadurch behoben werden, daß alle noch vor diesem Arbeitsplatz wartenden Operationen auf andere, weniger belastete Arbeitsplätze verteilt werden. Durch die Auslösung von Bewertung und Umplanung durch Ereignisse kann der Benutzer lernen, unter welchen Umständen in der realen Fertigung umgeplant werden muß. Der Ausfall einer einzigen Maschine wird i.d.R. keine Umplanung rechtfertigen, aber mehrere Ausfälle oder andere Umstände können eine Umplanung notwendig machen. Zur Durchführung der Umplanung wird in DYBBS die in WIZARD angebotene Funktionalität verwendet.

Analyse der Simulationsergebnisse

Nach der Durchführung einer Simulation stehen dem Benutzer mehrere Auswertungsfunktionen zur Verfügung, die verschiedene Aspekte des Simulationsergebnisses betrachten. Eine Möglichkeit ist die Bewertung des Simulationsablaufs durch WIZARD. Dabei wird anstelle des ursprünglichen Belegungsplans der Simulationsablaufplan durch die Constraints von WIZARD bewertet. Da die Constraints auch bei der Planerstellung in WIZARD verwendet werden, kann ein Vergleich der beiden Pläne aufgrund dieser Bewertungen stattfinden. Bei der Bewertung des gesamten Simulationsablaufplans stehen dem Benutzer ebenso wie bei der Umplanung oder Bewertung während der Simulation verschiedene Szenarien zur Verfügung.

Die zweite Analysemöglichkeit in DYBBS besteht in der Verwendung der Visualisierungsmodule von WIZARD, mit denen der durch die Simulation entstandene Ablaufplan graphisch und tabellarisch dargestellt werden kann. Dabei kann die Auslastung der Ressourcen in verschiedener zeitlicher Auflösung in einer Auslastungstabelle (s. Anhang B.18) untersucht werden. In Abbildung 39 bis Abbildung 42 (S. 117f) sind Beispiele für die graphische Ausgabe der Simulationsergebnisse abgebildet.

Der Hauptaspekt bei der Analyse der Simulationsergebnisse liegt im Vergleich zwischen Simulationsablauf und Belegungsplan. Dazu steht dem Benutzer eine vergleichende Bewertung der beiden Pläne zur Verfügung. Diese Bewertungen berechnen die Unterschiede anhand statischer Kriterien wie z.B. der DLZ oder dem Puffer zwischen Endtermin und ST einer Operation. Dabei wird für jede Operation jeweils die Differenz zwischen Plan und Simulation ermittelt und diese Unterschiede über alle Operation statistisch ausgewertet. Die Auswertung läßt sich auch nach einzelnen Ressourcen oder Aufträgen getrennt ausführen, wobei dabei allerdings die statische Aussagekraft geringer wird, da weniger Operationen als Grundlage dienen.

Die Anwendung statischer Bewertungskriterien ist für die Analyse nach einem Simulationslauf angebracht, da im Gegensatz zur Bewertung während eines Simulationslaufes zwei vollständige Belegungspläne vorliegen. Insgesamt wurden in DYBBS ca. 10 vergleichende Bewertungen implementiert, die die üblichsten PPS-Analysen darstellen. Dazu gehören die Auslastung von Ressourcen, Bearbeitungs-, Warte- und Durchlaufzeit von Aufträgen und Operationen, der Vergleich der Start- und Endtermine sowie die Puffer zwischen FT und Starttermin oder Endtermin und spätestetem Endtermin.

Die Ergebnisse können tabellarisch und in Diagrammen angezeigt sowie zur weiteren Untersuchung mit einem externen Werkzeug in eine Datei geschrieben werden. Bei Bedarf können leicht weitere Methoden zur Vergleichsbewertung hinzugefügt werden, wozu allerdings Kenntnisse der WIZARD- und DYBBS-Schnittstellen zum Zugriff auf die Daten notwendig sind. In Abschnitt 7.2 wird ein Beispielmodell vorgestellt, das diese vergleichende Bewertungen verwendet.

6.4. Zusammenfassung

In diesem Kapitel wurde eine erläuternde Einführung in das implementierte Simulationswerkzeug DYBBS gegeben. DYBBS ist als zusätzliches Modul zum Dispositionswerkzeug WIZARD implementiert und bietet daher viele Möglichkeiten für die Simulation von Fertigungssystemen anhand von PPS-Daten, die in anderen Simulationsumgebungen nicht direkt vorhanden sind. Der Kern von DYBBS besteht aus einer Implementierung des DES-Algorithmus, der mit einer graphischen Modellierungsmethode verbunden ist. Die Modellierung erfolgt hierarchisch nach den einzelnen Objekten der PPS-Daten. Zur Modellierung der einzelnen Elemente steht mit dem Zustandsgraph, einer Erweiterung von Ereignisgraphen, eine Methode zur Verfügung, die das Verhalten der Elemente intuitiv und übersichtlich darstellbar macht. Durch die Einbindung von Regeln zur Steuerung des Verhaltens kann der Benutzer beliebig komplexe Modelle entwerfen. Durch die Parametrisierung von Regeln und der Definition von speziellen Attributen für einzelne Elemente können Teilmodelle gut wiederverwendet und mit geringen Änderungen auf andere Elemente übertragen werden.

Da der Schwerpunkt von DYBBS in der Simulation von Fertigungssystemen im Zusammenhang mit der Bewertung von Belegungsplänen liegt, wurden auf den Simulationskern aufbauend verschiedene Teilmodule implementiert, die eine enge Anbindung der Simulation an WIZARD ermöglichen. Die Integration umfaßt die Modellbildung aus PPS-Daten, die Bewertung und Umplanung während eines Simulationslaufs und die vergleichende Bewertung von Simulationslauf und Belegungsplan nach der Simulation. Es wurden verschiedene Bewertungsfunktionen implementiert, wobei einfach weitere Kriterien hinzugefügt werden können.

7. Evaluation

In diesem Kapitel wird eine Evaluation von DYBBS durchgeführt, wobei der Schwerpunkt der Evaluation auf dem Aufzeigen der Möglichkeiten von DYBBS liegt. Es werden verschiedene Simulationsmodelle vorgestellt, die jeweils einen Aspekt von DYBBS genauer aufzeigen. Durch diese Evaluation wird die grundsätzliche Anwendbarkeit von DYBBS auf eine Vielzahl von Problemstellungen innerhalb der Fertigungssteuerung demonstriert. Es wird keine Evaluation anhand einer komplexen, an realen Daten und der Problemstellung einer realen Fertigung ausgerichteten Simulation durchgeführt.

In Abschnitt 7.1 wird ein einfaches System aus der Warteschlangentheorie simuliert. Da dieses analytisch untersuchbar ist, kann anhand dieses Systems eine Verifikation des Statistikmoduls und des Simulationskerns (s. Abschnitt 6.3) erfolgen. Aufbauend auf einer korrekten Grundlage werden daraufhin in den Abschnitten 7.2 bis 7.5 vier komplexere Simulationsmodelle vorgestellt, die jeweils einen Aspekt der Simulation von Fertigungsprozessen genauer betrachten. In Abschnitt 7.2 wird gezeigt, wie in DYBBS die Erstellung eines Ressourcenbelegungsplanes durch Simulation erfolgen kann. Das Modell in Abschnitt 7.3 konzentriert sich auf das Zusammenwirken von Maschinen und Personal in einer Fertigung. Abschnitt 7.4 demonstriert die Fähigkeiten von DYBBS, während der Simulation dynamisch Umplanungen durchzuführen. Die Überprüfung von Sicherheitspuffern bei der Planerstellung in einer störanfälligen Fertigungsumgebung steht im Mittelpunkt des Modells von Abschnitt 7.5.

7.1. M / M / 1 - Warteschlangenmodell

Als erstes Evaluationsbeispiel wird das aus der Warteschlangentheorie bekannte System M/M/1-∞ betrachtet. Die Notation bedeutet, daß an einer Bedienstation der Kapazität *n=1* mit unendlich großem Warteraum Anforderungen eintreffen, deren Zwischenankunftsverteilung *A* (erstes M, für Markov) negativ-exponentiell ist, und die in negativ-exponentiell verteilten Bedienzeiten *B* (zweites M) bearbeitet werden. Das M/M/1-∞ stellt ein sehr einfaches System dar, das analytisch gelöst werden kann, weshalb es als erstes Beispiel gewählt wurde, um eine Verifikation (s. Abschnitt 2.4) des Statistik- und DES-Moduls von DYBBS durchzuführen. Nachdem die grundsätzliche Korrektheit des Simulationskerns aufgezeigt wird, können ausgehend von dieser Grundlage komplexe Modelle simuliert werden. Für das System M/M/1-∞ wurden folgende Parameter, wobei weder die Größenordnung der Zeiteinheit noch die Werte eine besondere Bedeutung haben, gewählt:

- Ankunftsrate: $\lambda=0.015\ s^{-1}$ (d.h. mittlere Zwischenankunftszeit *E[A]=66.7s*)
- Bedienrate: $\mu=0.020\ s^{-1}$ (d.h. mittlere Bedienzeit *E[B]=50.0 s*)

7.1.1. Analytische Lösung

Aus der Warteschlangentheorie sind folgende Eigenschaften für das System M/M/1-∞ bekannt (s. [Tra96]):

- Ankunftsverteilungsfunktion $A(t) = 1 - e^{-\lambda t}$. Insbesondere gelten für die Korrelationskoeffizienten: $c_A=1$ und $c_B=1$.
- Auslastung: $\rho = \frac{\lambda}{\mu \cdot n} = 75\%$
- Mittlere Warteschlangenlänge: $\Omega = E[X_W] = p_W \cdot \frac{\rho}{1-\rho} = 2.25$ Anforderungen
- Mittlere Wartezeit (bzgl. aller Anforderungen): $E[W] = \frac{\Omega}{\lambda} = 150s$

7.1.2. Simulation

Das DYBBS-Modell besteht aus 4 Elementen: 1 Quelle, 1 Senke, 1 Arbeitsplatz und 1 Simulator. Für eine Simulationsdauer von *t=86400s* und *n=10* Simulationsläufen wurde folgendes Ergebnis ermittelt (s. Abbildung 22):

⇒ Die mittlere Zwischenankunftszeit, d.h. die zeitlichen Abstände zwischen der Erzeugung von Anforderungen an der Quelle, (Mittelwert[Arrival,arbeitend]=`00:01:07`) und die mittlere Bedienzeit (Mittelwert[Server,arbeitend]=Mittelwert[Dummy-Operation,arbeitend]=`00:00:50`) entsprechen den Vorgaben.

⇒ Die mittlere Auslastung der Bedieneinheit (Relativ[Server,arbeitend]=`74.8%`) stimmt mit der errechneten Auslastung überein.

⇒ Im Durchschnitt wurden 1275 Anforderungen erzeugt.

⇒ Die mittlere Wartezeit (Mittelwert[Dummy-Operation,wartend]=`00:02:35`) übersteigt die berechnete Wartezeit nur geringfügig. Die mittlere Warteschlangenlänge (nicht im Bild) beträgt Ω = `2.28`.

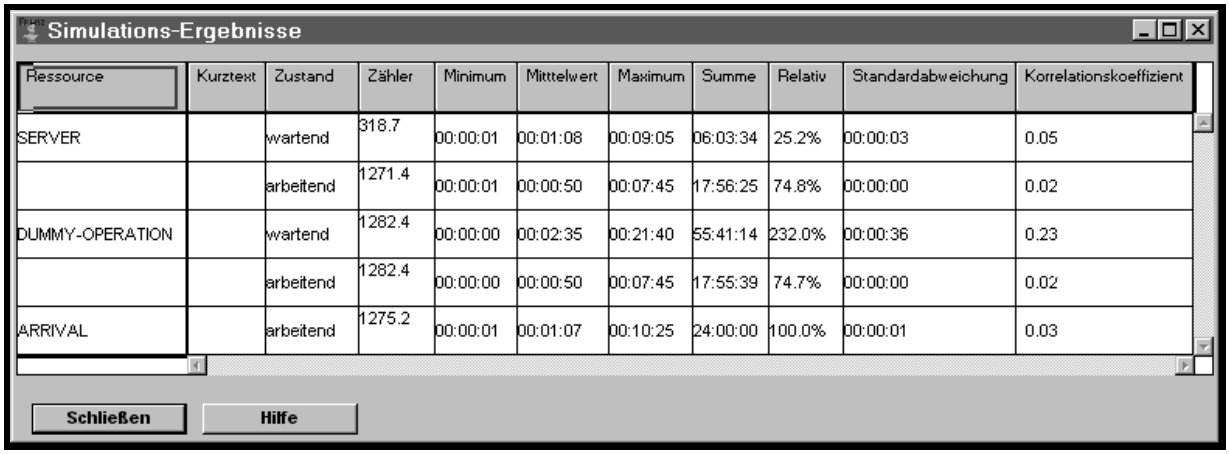


Abbildung 22: Evaluation - M/M/1-Wartesystem

Die Simulation liefert korrekte Ergebnisse für das Modell M/M/1-∞. Die aufgrund der implementierten Statistikmöglichkeiten machbaren Aussagen stimmen mit der analytischen Lösung überein, d.h. die Implementation des Statistikmoduls und des Simulationskerns kann als korrekt angesehen werden.

7.2. Scheduling in DyBBS

Im zweiten Evaluationsbeispiel wird demonstriert, wie in DYBBS Scheduling durch Simulation betrieben werden kann. Die Elemente der Simulation werden dabei aus PPS-Daten einer virtuellen Fertigung übernommen und mit einfachen Zustandsgraphen gekoppelt. Das Modell enthält keine stochastischen Komponenten, d.h. die Erstellung des Belegungsplan durch Simulation erfolgt deterministisch. Wie bereits in Abschnitt 4.2.2 beschrieben, erfolgt die Erstellung des Belegungsplans durch die Prioritätsregeln der Warteschlangen. In DYBBS sind ca. 10 einfache, lokale Prioritätsregeln (z.B. FIFO, kürzeste Bearbeitungszeit, Sortierung nach ST) implementiert, die aber bei Bedarf beliebig erweitert werden können.

Als Aufgabenstellung sollen vier Szenarien mit verschiedenen Prioritätsregeln simuliert und die entstandenen Belegungspläne mit dem Originalplan im Hinblick auf die DLZ der Aufträge und die Auslastung der Ressourcen untersucht werden. Ziel dabei ist es, die DLZ zu minimieren und eine hohe Auslastung zu erhalten. Gleichzeitig müssen die Endtermine der Aufträge eingehalten werden.

7.2.1. Modellbeschreibung

Die PPS-Daten dieses Evaluationsbeispiels haben folgenden Umfang:

- 12 Arbeitsplatzgruppen mit insgesamt 16 Einzelarbeitsplätzen
- 31 Aufträge mit insgesamt 190 Operationen
- die Planung erfolgt auf der Ebene der Arbeitsplatzgruppen

Für die Simulation ergeben sich folgende harten Bedingungen, die im Simulationsmodell berücksichtigt werden müssen:

- alle Operationen müssen bearbeitet werden
- eine begonnene Operation kann nicht unterbrochen werden
- keine Operation darf vor ihrem FT oder vor dem Ende ihrer Vorgänger gestartet werden
- jede Arbeitsplatzgruppe kann nur soviel Operationen gleichzeitig bearbeiten wie Einzelarbeitsplätze vorhanden sind[18].

Das Simulationsmodell enthält keine Zufallsquellen (z.B. Ausfälle), d.h. es ist deterministisch. Durch verschiedene Prioritätsregeln in den Warteschlangen der Arbeitsplatzgruppen können unterschiedliche Belegungspläne durch die Simulation erstellt werden. Dabei soll in diesem Beispiel der Einfluß der Prioritätsregeln auf die Durchlaufzeit untersucht werden. Insgesamt wurden vier Szenarien durchgespielt (s. Tabelle 5)

Szenario	Beschreibung
1	FIFO
2	kürzeste Bearbeitungsdauer zuerst
3	ST zuerst
4	gemischt (FT bis auf 2 Gruppen)

Tabelle 5: Scheduling durch Simulation - Szenarien

Im Szenario 4 arbeiten alle bis auf zwei Arbeitsplatzgruppen die ankommenden Operationen nach der Prioritätsregel „FT bestimmt die Reihenfolge“ ab. Die Arbeitsplatzgruppe (Rob-5010), in der unter der Strategie „FT zuerst“ die größte Warteschlangenlänge aufwies, wurde auf die Disziplin „kürzeste Bearbeitungsdauer“ gesetzt, da diese einen hohen Durchsatz garantiert. Die für die meisten Aufträge am Ende stehende Arbeitsplatzgruppe für die Endmontage (Mont-1200) wurde auf die Priorität „ST bestimmt die Reihenfolge“ gesetzt, da an dieser Stelle die Einhaltung der Endtermine der Aufträge vorrangig ist. Für das Simulationsmodell mußte ein Zustandsgraph für eine Quelle, die die Freigabe der Operationen zu ihrem FT steuert, und ein Zustandsgraph für die Arbeitsplatzgruppen erstellt werden. Diese benötigten insgesamt 10 Regeln. Die unterschiedlichen Prioritätsregeln in den Szenarien machen keine eigene Modellierung für die Arbeitsplatzgruppen notwendig, d.h. ein Teilmodell für die Arbeitsplatzgruppen konnte in allen vier Szenarien für alle Arbeitsplatzgruppen verwendet werden. Die entwickelten Teilmodelle für die Quelle und Arbeitsplatzgruppen sind unabhängig von den tatsächlichen PPS-Daten über Aufträge und Ressourcen und können daher sofort auf andere Daten übertragen werden. Die einzelnen Arbeitsplätze spielen in diesem Beispiel keine Rolle und brauchen daher nicht modelliert werden.

7.2.2. Simulationsergebnisse

Durch das Fehlen von Zufallseinflüssen in diesem Beispiel muß für jedes Szenario nur ein Simulationslauf durchgeführt werden. Aus den ersten Simulationsläufen war zu erkennen, daß ein Simulationszeitraum zwei Monaten ausreichend ist, wobei die gegebenen frühesten und spätesten Termine der Aufträge beachtet werden.

In allen vier Szenarien können die gegebenen Aufträge ohne Überschreitung eines ST eines Auftrags im Simulationszeitraum eingeplant werden. Alle vier Szenarien erhalten bei der Bewertung durch

[18] In diesem Beispiel haben alle Einzelarbeitsplätze eine Kapazität von 1.

WIZARD ähnliche Bewertungen wie der Originalplan, da nur eine oder zwei Operationen aus ihren Puffern fallen. Abbildung 23 zeigt die Auslastung für Szenario 4, die sich nicht wesentlich von den anderen Szenarien unterscheidet.

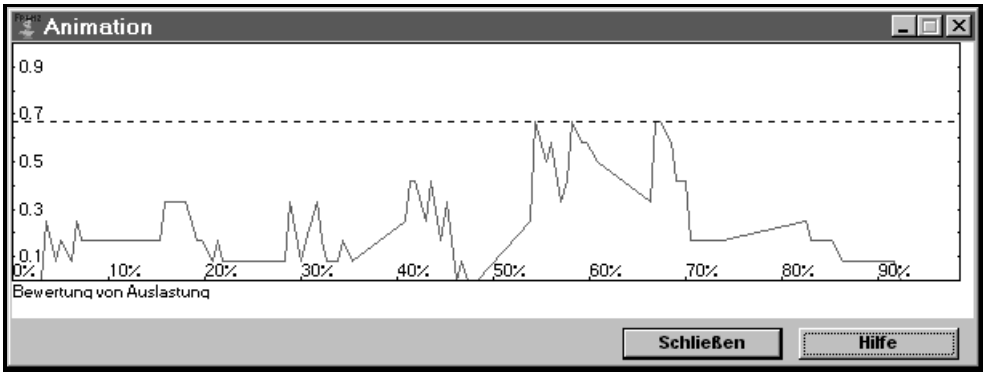


Abbildung 23: Auslastung für Szenario 4

Erklärung: Die X-Achse stellt den Simulationszeitraum dar. Die Y-Achse zeigt die Auslastung aller Ressourcen (als Anteil zwischen 0 und 1).

Bei der Betrachtung der DLZ (s. Tabelle 6) zeigt sich, daß Szenario 2 die größte Verbesserung gegenüber dem Originalplan bringt. In diesem Szenario werden die kürzeren Operationen bevorzugt, so daß diese eine Verbesserung ihrer DLZ auf Kosten von wenigen langen Operationen erfahren. Die Benachteiligung der längeren Operationen fällt jedoch nicht so stark ins Gewicht. Die maximale Verschlechterung für eine Operation gegenüber dem Originalplan war in diesem Szenario nicht größer als in den anderen Szenarien. Die speziellen Einstellungen in Szenario 4 führen zu einer geringfügigen Verbesserung gegenüber den Szenarien 1 und 3. Als Resultat kann festgehalten werden, daß ein Zeitraum von zwei Monaten ausreicht, ohne daß es zu Terminschwierigkeiten kommt. Es kann eine durchsatzorientierte Planung wie in Szenario 2 empfohlen werden, da diese für die gegebenen Daten zu keinen Terminschwierigkeiten führt. Weiterhin zeigte sich in allen vier Szenarien, daß eine Arbeitsplatzgruppe (Dreh-3080) eine besonders große Verbesserung der DLZ erzielen konnte. Diese betrug je nach Szenario zwischen 7h und 10h. Dies zeigt, daß die Einplanung durch WIZARD an einzelnen Stellen optimiert werden kann.

Bewertungskriterium	Szenario 1	Szenario 2	Szenario 3	Szenario 4
größte Verbesserung der DLZ	-37:27h	-28:23h	-27:49h	-27:49h
mittlere Änderung der DLZ (auftragsbezogen)	-00:08h	-00:36h	-00:13h	-00:16h
mittlere Änderung der DLZ (ressourcenbezogen)	-00:13h	-00:42h	-00:12h	-00:19h
größte Verschlechterung der DLZ	+33:49h	+34:01h	+27:23h	+34:01h
Wizard-Bewertung (verletzte Constraints,Originalplan = 5.42)	5.42	10.66	5.42	10.66

Tabelle 6: Ergebnisse des Scheduling-Beispiels

Das vorgestellte Beispiel zeigt, daß in DYBBS durch ein einfaches Simulationsmodell, das auf der hohen Abstraktionsebene der Arbeitsplatzgruppen arbeitet, Scheduling betrieben werden kann. Die entstehenden Belegungspläne können die harten Constraints, die im Modell vorhanden sind, nicht verletzen und erzeugen somit Belegungspläne, die den Grundanforderungen genügen. Durch die Angabe von Prioritätsregeln kann der Belegungsplan auf ein bestimmtes Ziel hin optimiert werden. Dabei können Erkenntnisse gewonnen werden, die zwar nicht allgemeingültig sind, aber in der Praxis auf den Einzelfall der aktuellen Daten anwendbar sind. Die Übertragbarkeit der Teilmodelle erlaubt es, innerhalb von wenigen Minuten eine Simulation für das Schedulingproblem aufzubauen und durchzuführen.

7.3. Maschinen-Personal-Modell

Als zweites Evaluationsbeispiel wird das Zusammenwirken von Arbeitsplätzen und Werkern simuliert. Dabei steht die Fragestellung, wieviele Werker benötigt werden, um eine vorgegebene Menge von Aufträgen auf bestimmten Arbeitsplätzen zu bearbeiten, und welchen Einfluß die Anzahl der Werker auf die

Bearbeitungszeit der Operationen besitzt. Im Gegensatz zum vorhergehenden Evaluationsbeispiel wird an dieser Stelle ein detaillierteres Simulationsmodell entwickelt, das sowohl auf der Arbeitsplatz- als auch auf der Personalseite eine Aufteilung von Gruppen- und Einzelressourcen besitzt.

7.3.1. Modellbeschreibung

Als Ausgangsdaten liegen die gleichen Daten wie im vorigen Beispiel, d.h. ein Ressourcenbelegungsplan auf der Ebene der Arbeitsplatzgruppen. Zusätzlich kommen in diesem Beispiel auch Personalressourcen zum Einsatz. Diese arbeiten abwechselnd in zwei Schichten (Früh- und Spätschicht).

Die auf der Planungsseite vorhandene Zuordnung auf Arbeitsplatzgruppenebene wird im Simulationsmodell verfeinert, indem die einzelnen Operationen auf Einzelarbeitsplätzen bearbeitet werden. Dazu wird ein Teilmodell für Arbeitsplatzgruppen und ein Teilmodell für Einzelarbeitsplätze benötigt. Das Teilmodell für die Arbeitsplatzgruppen hat ein sehr einfaches Verhalten. Für jede Operation, die in dieser Gruppe bearbeitet werden soll, wird ein Einzelarbeitsplatz bestimmt und die Operation wird an den Einzelarbeitsplatz weitergeleitet. Das Teilmodell für die Arbeitsplatzgruppe dient somit nur zur Verteilung der Operationen auf Einzelarbeitsplätze. Die Verteilung erfolgt im vorliegenden Fall nach der Anzahl der vor einem Arbeitsplatz wartenden Operationen, wobei aber durch Änderung einer Regel beliebige Verteilungsstrategien eingebaut werden können.

Ein Einzelarbeitsplatz benötigt zur Durchführung der Bearbeitung einen Werker. Daher meldet er bei der gerade aktiven Schicht seinen Bedarf an. Diese nimmt die Anforderung entgegen und bestimmt einen Werker, dem der Arbeitsplatz zugewiesen wird. Die Verteilung erfolgt an den Werker, bei dem im Moment die wenigsten Anforderungen vorliegen, wobei die Werker keinen Einschränkungen hinsichtlich ihrer Qualifikation unterliegen.

Ein Werker, der gerade nicht beschäftigt ist und eine Anforderung erhält, beginnt mit der Bearbeitung der Operation auf dem Arbeitsplatz. Kann er die Operation innerhalb seiner Schicht beenden, so ist die Operation beendet, die Nachfolgeoperation kann angestoßen werden und der Werker kann eine neue Anforderung beginnen. Tritt aber der Fall ein, daß die Operation nicht innerhalb der Schicht beendet werden kann, so arbeitet der Werker bis zu seinem Schichtende an dieser Operation und übergibt die angefangene Arbeit an die nächste Schicht. Dazu muß bestimmt werden, welchen Anteil an der Operation der Werker erledigt hat und wie groß die Restarbeit für diese Operation ist. Am Schichtende erhält die nächste Schicht die angefangene Operation sowie alle Anforderungen, die der Werker zwar erhalten, aber nicht begonnen hat. Die Schicht beginnt damit, daß alle Operationen wie oben auf einzelne Werker verteilt werden und diese mit der Bearbeitung anfangen bzw. die Bearbeitung fortsetzen. Können die Anforderungen wiederum nicht in dieser Schicht abgeschlossen werden, so erhält die erste Schicht wieder Operationen zugewiesen usw. Da innerhalb der Schichten unterschiedlich viele Werker beschäftigt sein können, können unter Umständen angefangene Operationen nicht sofort fortgesetzt werden, sondern sind solange unterbrochen, bis ein Werker zur Bearbeitung zur Verfügung steht.

Das Modell enthält somit fünf Teilmodelle (Quelle, Arbeitsplatz- und Personalgruppe, Arbeitsplatz und Werker), die im Zusammenspiel den Ablauf der Fertigung abbilden. Insgesamt wurden in diesem relativ komplexen Modell ca. 20 Regeln benötigt.

7.3.2. Simulationsergebnisse

Wie im vorigen Evaluationsbeispiel wurde ein Simulationszeitraum von zwei Monaten gewählt. Da für die betrachtete Aufgabenstellung die Personalressourcen den Engpaß in der Fertigung darstellen sollen, müssen weniger Werker als Einzelarbeitsplätze vorhanden sein. Daher wurden in die Simulation zwischen 5 und 11 Werker aufgenommen, die möglichst gleichmäßig auf die beiden Schichten verteilt werden. Als Ergebnis der Simulation können für die gegebenen Ausgangsdaten folgende Erkenntnisse gewonnen werden:

- Bei weniger als 10 Werkern können nicht alle Aufträge pünktlich beendet werden.
- Die Auslastung steigt mit sinkender Anzahl der Werker, allerdings erhöhen sich die Durchlaufzeiten gegenüber den Vorgabedaten erheblich. Die durchschnittliche Dauer einer Operation anhand der PPS-Daten liegt bei 7:15h. Bei 8 Werkern ergibt sich bereits eine Erhöhung um ca. 40% durch zusätzliche Liegezeiten.
- Bei 5 Werkern können im gegebenen Terminrahmen nicht mehr alle Operationen bearbeitet werden.

Abbildung 24 zeigt die Änderung von Auslastung, Bearbeitungsdauer, Anzahl verspäteter Aufträge und die Bewertung durch WIZARD in Abhängigkeit von der Anzahl der eingesetzten Werker. Insgesamt kann für die gegebene Datenlage der Einsatz von 10 Werkern empfohlen werden, die jedoch dann nur zu ca. 50% ausgelastet sind.

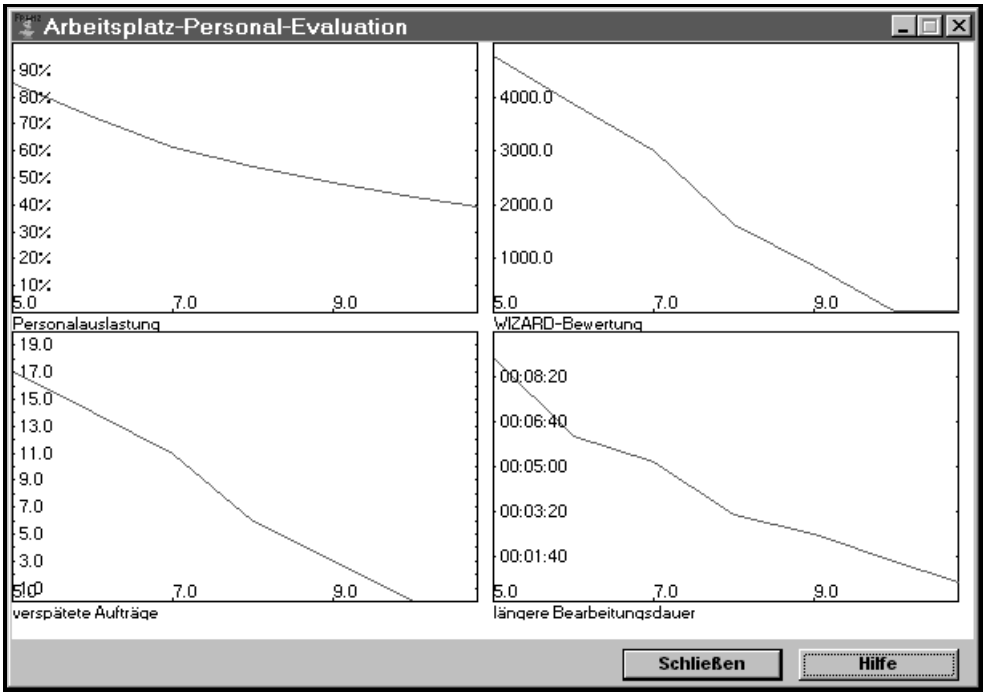


Abbildung 24: Ergebnisse des Arbeitsplatz-Personal-Beispiels

Erklärung: In allen vier Graphiken stellt die X-Achse die Anzahl der eingesetzten Werker dar. Die Y-Achse zeigt in der jeweiligen Graphik die Auslastung der Werker (in Prozent), die Bewertung durch WIZARD (in Strafpunkten), die Anzahl der verspäteten Aufträge und die Verlängerung der Bearbeitungszeit durch Liegepausen (in Stunden) dar.

Dieses Modell kann an vielen Stellen erweitert werden. So können zum Beispiel die Krankheit von Werkern modelliert oder Maschinenausfälle eingebaut werden. Ebenso könnten den Werkern verschiedene Fähigkeiten (z.B. welche Maschinen sie bedienen können) zugewiesen werden. [Tou93] stellt ein weiteres Werkermodell vor, in dem Werker Haupt- und Nebenbeschäftigungen nachgehen und Pausen einlegen.

Das implementierte Modell zeigt allerdings deutlich, daß in DYBBS auch komplexe Modelle, die verschiedene Ebene umfassen, modelliert werden können. Das gezeigte Beispiel der Abschätzung des Personalbedarfs für gegebene Aufträge unter Berücksichtigung einer krankheitsbedingten Kapazitätsschwankung kann gut zur Ergänzung der Personaldisposition (z.B. für Ferienzeiten und die Einstellung von Aushilfskräften) verwendet werden.

7.4. Eilaufträge einplanen

In diesem Evaluationsbeispiel werden während der Simulation zu einer gegebenen Menge von Aufträgen dynamisch weitere Aufträge erzeugt, eingeplant und simuliert. Dabei wird untersucht, ob das Bearbeiten

von Zusatzaufträgen durchgeführt werden kann, ohne daß die bereits vorhandenen Aufträge in Terminschwierigkeiten kommen. Zusätzlich können Maschinen ausfallen, was einen weiteren Unsicherheitsfaktor darstellt.

7.4.1. Modellbeschreibung

Wie in den letzten beiden Beispielen liegen PPS-Daten über 31 Aufträge vor, die auf 12 Arbeitsplatzgruppen disponiert sind. Die Simulation verwendet 29 Aufträge mit 170 Operationen als von Anfang an gegebene Aufträge. Die anderen beiden Aufträge werden als Vorlagen zur dynamischen Auftragsgenerierung benutzt. Dabei wird zu zufälligen Zeitpunkten einer der beiden Aufträge dupliziert, von WIZARD eingeplant und in der Simulation freigegeben. Die erzeugten Aufträge haben im Gegensatz zu den anderen Aufträgen keinen festen Terminrahmen.

Die Auftragsgenerierung wird durch ein Objekt der Elementklasse Welt durchgeführt. Dieses besitzt keine speziellen Eigenschaften und dient zur Modellierung von Ereignissen, die außerhalb des eigentlichen Fertigungssystems liegen. In diesem Fall werden durch das Weltobjekt die zufälligen Auftragseingänge an den Betrieb modelliert. Als typische Zwischenankunftsverteilung für die Auftragseingänge wurde eine negativ-exponentielle Verteilung gewählt. In diesem Modell wird während eines Simulationslaufs auf das Ereignis des Auftragseingangs eine Umplanung vorgenommen, bei dem sowohl der neue Auftrag eingeplant wird als auch andere Operationen so umgeplant werden, daß die neuen Aufträge möglichst gut eingepaßt werden können. Bei der Umplanung muß dabei berücksichtigt werden, daß bereits bearbeitete oder in der Bearbeitung befindliche Operationen fixiert sind, d.h. diese können nicht mehr umgeplant werden.

Die Modellierung der Fertigung verläuft analog zur Modellierung im Abschnitt 7.2, d.h. die Operationen werden auf der Ebene der Arbeitsplatzgruppen bearbeitet. Diese können je nach ihrer Kapazität mehrere Operationen gleichzeitig bearbeiten. Zusätzlich zum deterministischen Modell des Schedulingbeispiels wurde ein Ausfall von Maschinen eingefügt. Dieser findet ebenfalls auf der Arbeitsplatzgruppenebene statt, d.h. im Störungsfall sind alle im Moment bearbeiteten Operationen unterbrochen. Ein detaillierteres und realistischeres Modell könnte an dieser Stelle die Störung auf der Ebene der Einzelressourcen durchführen, wobei einer Arbeitsplatzgruppe nur ein Teil der vorhandenen Kapazität verloren geht. Die Abstände zwischen zwei Störungen und die Dauer der Störung wurden durch Normalverteilungen bzw. Dreiecksverteilungen dargestellt, wobei für alle Arbeitsplatzgruppen die gleichen Verteilungen gewählt wurden. Auch an dieser Stelle könnten in einem anspruchsvolleren Modell maschinenspezifische Ausfalldauern und -abstände, die aus Meßdaten gewonnen wurden, eingesetzt werden.

7.4.2. Simulationsergebnisse

Als Parameterwerte für das Modell wird willkürlich für die Auftragserzeugung eine exponentielle Verteilung mit Mittelwert $\lambda^{-1}=144h$ und für die Maschinenstörung eine normalverteilte Zwischenausfallsdauer mit Parametern $m=24h$, $\sigma=3h$ und eine dreiecksverteilte Reparaturdauer mit Parametern $a=2h$, $b=5h$, $c=7h$ für alle Arbeitsplatzgruppen gewählt (s. Anhang C), wobei die Zeitangaben für die Arbeitsplatzgruppen schichtplanabhängig gerechnet werden. Für die Simulation wurden 25 Läufe über einen Simulationszeitraum von 70 Tagen durchgeführt. Dabei zeigten sich folgende Ergebnisse:

- Im Durchschnitt werden *4.12* Aufträge erzeugt. Alle fest eingeplanten Aufträge werden im simulierten Zeitraum bearbeitet.
- In den meisten Läufen überschreiten zwei Aufträge ihren Terminrahmen, wobei meistens ein Auftrag nur eine geringfügige Terminüberschreitung aufweist. Die durchschnittliche Bewertung eines Simulationslaufs durch WIZARD liegt bei *480.75* Strafpunkten, wobei der Terminverzug eines Auftrags bei ausreichend hoher Überschreitung mit *256* Strafpunkten gewertet wird.

- Die Analyse des Restpuffers zwischen Endtermin und SET einer Operation zeigt, daß drei Aufträge (Demo23, Demo12, Demo01) für fast alle auftretenden Terminüberschreitungen verantwortlich gemacht werden können.

Als Ergebnis kann für dieses Simulationsmodell festgestellt werden, daß trotz der zusätzlich ankommenden Aufträge fast alle Aufträge ohne Terminverzug bearbeitet werden können. Aus der Simulation ist zu erkennen, für welche Aufträge Probleme zu erwarten sind, so daß der Disponent für diese Aufträge manuell eingreifen kann.

In diesem Simulationsbeispiel wird deutlich, welche Möglichkeiten DYBBS zur Umplanung während des Simulationslaufs bietet. Die Umplanung kann dabei so gestaltet werden, daß bestimmte Operationen auf jeden Fall umgeplant werden müssen (zusätzliche Aufträge), andere Operationen nicht mehr verschoben werden können (fixierte Operationen), während die übrigen Operationen nach den Möglichkeiten ihres Terminrasters verschoben können. Die Umplanung wird in diesem Beispiel ereignisgesteuert durch den Eingang eines Auftrags ausgelöst. Ein anderer Auslösungsmechanismus steht in der dynamischen Bewertung des Simulationsablaufs zur Verfügung. Abbildung 25 zeigt den in Abschnitt 4.6 vorgestellten Pass Error für einen der durchgeführten Simulationsläufe als Beispiel für ein dynamisches Bewertungskriterium. Ebenso könnte in diesem Modell eine einmal erzeugte Ereigniskette für die Zeitpunkte der Auftragseingänge verwendet werden und diese auf verschiedene Umplanungsstrategien[19] angewendet werden.

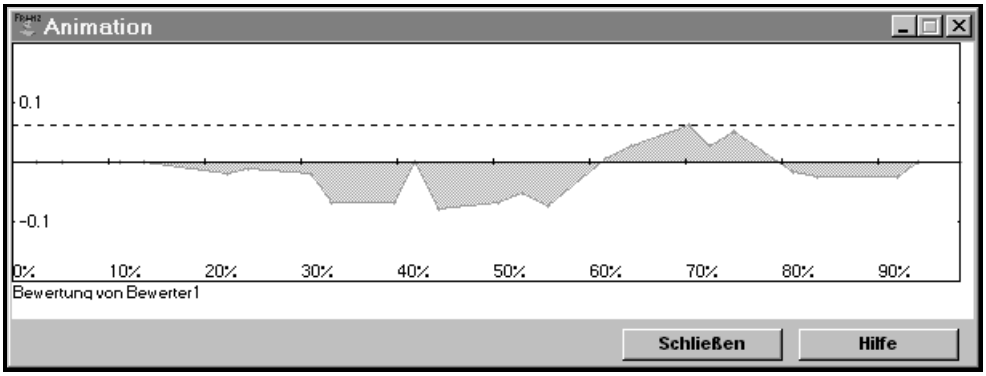


Abbildung 25: Eilaufträge einplanen - Pass Error

Erklärung: Die X-Achse stellt den Simulationszeitraum dar. Negative Y-Werte bedeuten einen Terminverzug, positive einen Terminvorsprung. Der quantitative Wert ist als Verhältnis der verspäteten Operationen im Bezug zu allen eingeplanten Operationen zu sehen.

7.5. Parameterlauf mit Sicherheitspuffern

Das folgende Simulationsbeispiel stellt einen Parameterlauf dar, bei dem für ein störanfälliges Fertigungsumfeld ein geeigneter Sicherheitsspielraum für die Planung gefunden werden soll, so daß die Störungen in der Fertigung während der Planerstellung ausreichend berücksichtigt werden. Die Berücksichtigung soll dadurch erfolgen, daß zur tatsächlichen Bearbeitungsdauer einer Operation ein zu der Dauer proportionaler Sicherheitsspielraum aufgeschlagen wird.

7.5.1. Modellbeschreibung

Dem Simulationsmodell liegen die in bereits in den vorigen Beispielen verwendeten PPS-Daten zugrunde, d.h. es werden insgesamt 190 Operationen auf 12 Arbeitsplatzgruppen bearbeitet. Planerstellung und Simulation erfolgen auf der Ebene der Arbeitsplatzgruppen.

Ausgehend von dem in Abschnitt 7.4 beschriebenen Simulationsmodell wird das Modell eines Arbeitsplatzgruppe mit einer Störkomponente verwendet. Arbeitsplatzgruppen können zu beliebigen Zeitpunkten während einer Bearbeitung ausfallen und müssen daraufhin repariert werden. Sowohl die Ab-

[19] Verschiedene Umplanungsstrategien sind z.Z. noch nicht in WIZARD implementiert.

stände zwischen zwei Ausfällen als auch die Dauer der Reparatur sind zufallsabhängig. Ein inaktiver Arbeitsplatz kann nicht ausfallen. Der Ausfall einer Arbeitsplatzgruppe ist so modelliert, daß der Ausfall die gesamte Kapazität der Arbeitsplatzgruppe betrifft, d.h. für den Ausfallzeitraum steht keine Kapazität zur Verfügung. Ein detaillierteres Modell könnte an dieser Stelle die Bearbeitung in einer Arbeitsplatzgruppe auf der Ebene der Einzelarbeitsplätze beschreiben, so daß ein Ausfall eines Einzelarbeitsplatzes nur eine Verminderung der Kapazität zur Folge hat.

Aufgrund fehlender Daten wird für alle Arbeitsplätze eine einheitliche Störcharakteristik festgelegt. Dabei wird die Dauer zwischen zwei Ausfällen durch eine gleichmäßige Verteilung mit den Grenzen (7h, 15h) und die Dauer der Reparatur durch eine Dreiecksverteilung mit den Parametern (5h, 8h, 9h) modelliert (s. Anhang C). Diese übertriebenen Annahmen wurden zum Zweck der Verdeutlichung der Simulationsergebnisse gewählt und können im konkreten Fall durch individuelle, aus der Betriebsdatenerfassung verfügbaren Schätzungen ersetzt werden.

Zur Durchführung des Parameterlaufs wird eine Experimentkontrolle (s. Anhang B.16) benötigt, die ausgehend vom Parameter des prozentuellen Sicherheitsaufschlags zur Operationsdauer für jeden Simulationslauf einen neuen Belegungsplan erzeugt, der die veränderten Bearbeitungsdauern enthält. Das Simulationsmodell verwendet jedoch für die simulierten Bearbeitungsdauern die Originalwerte. Für jeden so erzeugten Belegungsplan werden aufgrund der zufälligen Störkomponente mehrere Simulationsläufe durchgeführt und deren Ergebnisse mit dem Plan verglichen. Aus diesem Vergleich für jeden Parameterwert kann derjenige Sicherheitsaufschlag für die Planerstellung gefunden werden, der am ehesten dem Ablauf in der Simulation entspricht.

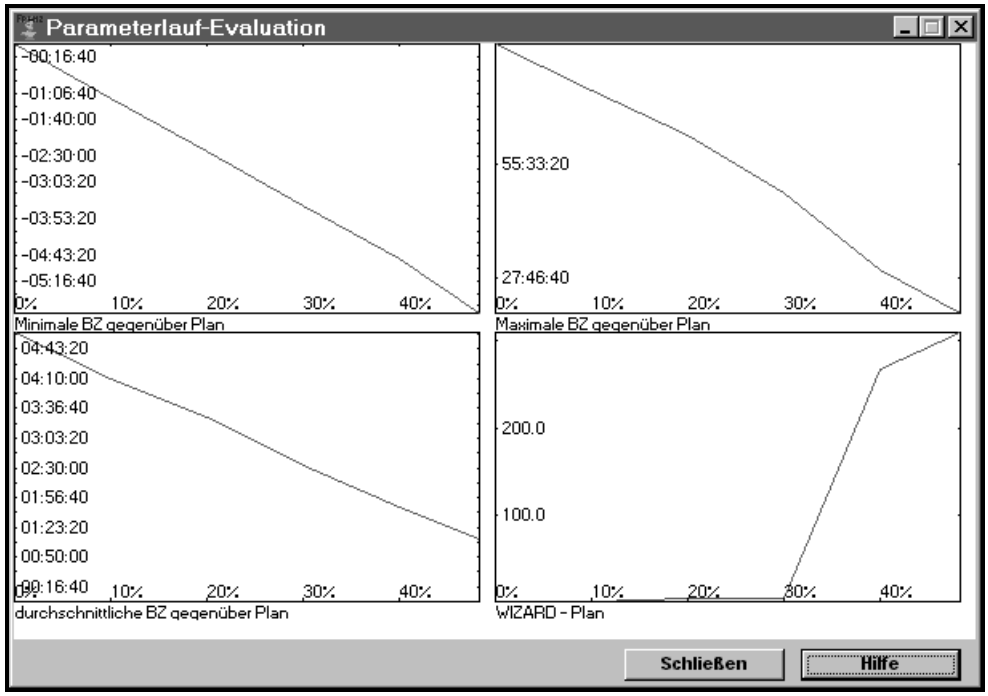


Abbildung 26: Evaluation - Parameterlauf

Erklärung: Die X-Achse ist in allen Graphiken der verwendete Sicherheitszuschlag. Die Y-Achse zeigt die minimale, maximale und durchschnittliche Veränderung der Berarbeitungszeiten (in Stunden) sowie die Bewertung durch WIZARD (in Strafpunkten).

7.5.2. Simulationsergebnisse

Als Parameterwerte werden Aufschläge zwischen 0% und 50% der Bearbeitungsdauer in 10%-Schritten verwendet. Für jeden Parameterwert müssen aufgrund der Zufallsabhängigkeit der Simulation mehrere

Simulationsläufe durchgeführt werden. Ein Wert von 5 Simulationsläufen erwies sich als ausreichend[20]. Das Ziel der Untersuchung war es, einen Sicherheitsaufschlag zu finden, der die pünktliche Terminierung aller Aufträge zuläßt und weitgehende Übereinstimmung von geplanten und simulierten Bearbeitungszeiten erreicht. Dabei ergaben sich die folgenden Ergebnisse:

- Falls kein Sicherheitsaufschlag vorgenommen wird, liegt die durchschnittliche Abweichung der simulierten Bearbeitungsdauer gegenüber der geplanten bei ca. +5h (ca. +68%).
- Die maximale Abweichung (d.h. das Maximum über alle Operationen und Läufe, der worst case) lag bei sehr hohen +84h (die größte Originaldauer beträgt 110h). Diese sehr hohe Abweichung liegt darin begründet, daß eine Operation mehrmals durch Störungen unterbrochen werden kann und daß die Störungen im Modell die ganze Arbeitsgruppe betreffen. Bei Einführung von Sicherheitszuschlägen konnte die maximale Abweichung stark gesenkt werden.
- Ab einem Sicherheitsaufschlag von 40% konnten nicht mehr alle Aufträge pünktlich terminiert werden, d.h. ein solcher Zuschlag behindert die Planung.
- Aufgrund der hohen Störanfälligkeit und Reparaturdauer sind im Durchschnitt die simulierten Durchlaufzeiten auch bei hohen Sicherheitszuschlägen größer als die geplanten. Dies liegt v.a. darin begründet, daß ein kleiner Teil der Operationen sehr große Verzögerungen hatte.

Abbildung 26 zeigt die Ergebnisse graphisch. Als Interpretation der Simulationsergebnisse ist festzuhalten, daß bei den gegebenen Annahmen ein Sicherheitsaufschlag von 30-40% die beste Lösung darstellt, da dadurch auf der einen Seite die Planung ohne Probleme arbeiten kann, wohingegen größere Sicherheitszuschläge im gegebenen Modell nur geringe Verbesserungen bei der Annäherung zwischen Simulations- und Planbearbeitungszeiten, v.a. für den Durchschnittlichsfall, zeigen.

7.6. Zusammenfassung

Die vorgestellten Beispielmodelle stellen eine Übersicht dar, welche Problemstellungen in DYBBS behandelt werden können, wie die Ergebnisse der Simulation repräsentiert werden und welche Interpretationen daraufhin gezogen werden können. Die vorgestellten Simulationsbeispiele dienen zum einen der Verifikation von DYBBS, indem wie in Abschnitt 7.1 die Simulation eines einfachen Warteschlangenmodells mit der analytischen Lösung verglichen wird. Zum anderen zeigen die Beispiele, daß DYBBS ein geeignetes Hilfsmittel innerhalb einer PPS-Umgebung sein kann, da es die Untersuchung vielfältiger Fragestellungen ermöglicht. Um einen guten Überblick über die Möglichkeiten von DYBBS zu geben, wurde die Evaluation mehrerer kleinerer Beispiele mit virtuellen Daten der Anwendung an einem komplexen Beispiel eines realen Fertigungsbetriebs mit entsprechenden Daten vorgezogen.

Die Erstellung eines Simulationsmodells mit Hilfe der vorhanden Regeln und Teilmodelle (s. Anhang A) ist auch ohne Systemkenntnisse über DYBBS und WIZARD möglich. Wenn es um die Erstellung neuer Teilmodelle oder die Untersuchung einer neuen Fragestellung geht, müssen durch den Modellierer neue Funktionen (z.B. in Auswirkungen oder Experiment-Ablaufkontrollen) programmiert werden, die sowohl Kenntnisse über die Programmiersprache LISP als auch über die Schnittstellen von WIZARD und DYBBS verlangen. Daher kann das Erstellen eines komplett neuen Simulationsmodells i.d.R. nur durch einen Wissensingenieur, nicht aber durch den Experten erfolgen. Die Verwendung vorhandener Teilmodelle ist hingegen auch ohne Kenntnisse über die Interna des Systems möglich. Insgesamt ist festzustellen, daß die vorgestellten Evaluationsbeispiele die Anwendbarkeit und den Nutzen von DYBBS hinreichend demonstrieren.

[20] Bei weiteren Experimenten mit z.B. 20 Simulationsläufen ergaben sich nur unwesentliche Änderungen zu den vorgestellten Durchnittswerten, aber eine erhöhte statistische Sicherheit und größere Simulationsdauern.

8. Zusammenfassung, Bewertung und Ausblick

In diesem Kapitel wird eine kurze Zusammenfassung über die Arbeit gegeben. Dabei wird sowohl auf die erarbeiteten theoretischen Grundlagen von Simulation und Disposition als auch auf die implementierte Simulationsshell DYBBS eingegangen. Die erreichten Ergebnisse werden einer Bewertung unterzogen. Abschließend soll auf mögliche Erweiterungen eingegangen werden.

8.1. Zusammenfassung

Die Zielsetzung dieser Arbeit war die Erstellung eines Simulationssystems zur Unterstützung der Feinplanung in PPS-Systemen. Daher wird neben der Analyse bestehender Simulationsansätze eine Untersuchung der in PPS-Systemen vorhandenen Daten, Methoden und Einsatzgebiete für Simulation benötigt.

8.1.1. PPS-Systeme

Als Fertigungsumfeld wird eine auftragsorientierte Werkstattfertigung angenommen, bei der weniger die globale Fertigungsplanung, sondern die zeitlich lokale Disposition im Vordergrund steht. Die planungsrelevanten Daten in PPS-Systemen betreffen die vorhandenen Ressourcen – Arbeitsplatz, Personal, Material und Werkzeug – und die Fertigungsstrukturen – Aufträge und Operationen. In einem Ressourcenbelegungsplan werden die Fertigungsstrukturen als Nachfrager den Ressourcen als Anbietern zugeordnet. Die Bewertung von Belegungsplänen erfolgt dabei durch Kennzahlen, die sich an den vier Grundzielen von PPS-Systemen – Auslastung, Durchlaufzeit, Lagerkosten und Termintreue – orientieren.

Diese Bewertungen können allerdings keinen Beitrag zur Robustheit des Planes liefern. Unter der Robustheit eines Planes wird dabei die Übereinstimmung zwischen Plan und Ausführung verstanden. Da der Produktionsprozeß zufälligen Schwankungen wie Maschinenstörungen unterliegt, stimmen die geplante und tatsächliche Produktion nicht überein. Eine Möglichkeit, diese stochastischen Schwankungen bei der Planung zu berücksichtigen, besteht in der Einführung von Sicherheitsgrößen, die allerdings oft nur geschätzt werden können. Durch die Modellierung des Produktionsprozesses unter Einbeziehung zusätzlichen Wissens über den Produktionsprozeß können jedoch durch Simulation quantitative Aussagen über die Abweichungen zwischen Plan und Ablauf und die Größe von Ausgleichsparametern gewonnen werden. Dazu werden dynamische Bewertungskriterien benötigt, die die Abweichungen zwischen Plan und Ablauf während der Produktion messen. Ein Beispiel für eine dynamische Bewertung ist die Messung der Verspätung, die der reale Ablauf zu einem Zeitpunkt gegenüber dem Plan zeigt. Überschreitet die Verspätung eine bestimmte Grenze, so kann der Plan durch Umplanung an den realen Ablauf angepasst werden. Diese Einsatzmöglichkeit für dynamische Bewertungen kann sowohl auf den realen Produktionsablauf als auch auf Simulationen angewendet werden. Durch die Simulation können vorausschauend für den aktuellen Zustand des Produktionsprozesses mögliche Engpässe und Probleme erkannt und diese in der Disposition berücksichtigt werden. Dabei können spezielle Frage- und Problemstellungen wie die voraussichtliche Termintreue bei zusätzlichen Sonderaufträgen gezielt untersucht werden, was ohne Simulation nur sehr schwer möglich ist, so daß mit der Simulation dem Disponenten ein zusätzliches Hilfsmittel bei der Erstellung von Ressourcenbelegungsplänen zur Verfügung steht.

Während bestehende Simulationssysteme sehr gut zur Fertigungsplanung, insbesondere für die Probleme des Materialtransports, geeignet sind, exisieren für die Unterstützung der Disposition oft nur grundlegende Module wie eine Datenbankschnittstelle.

8.1.2. Simulation

Neben Klassifikation und Konstruktion kann Simulation als eine der grundlegenden Problemlösungstypen angesehen werden ([Pup90]). Dabei wird unter Simulation die Abbildung eines zu untersuchenden Systems auf ein geeignetes Modell und die Durchführung von Experimenten am Modell verstanden. Da

das Modell eine Abstraktion des Systems darstellt, müssen die am Modell gewonnenen Ergebnisse auf ihre Konsistenz und Aussagekraft hin überprüft werden, bevor sie auf das ursprüngliche System übertragen werden.

Eine geeignete Klassifikation für den Problemlösungstyp Simulation kann durch charakteristische Eigenschaften des Modells gewonnen werden. Dabei muß bei der Analyse des Systems entschieden werden, welcher Modelltyp am besten zur Abbildung des Systems geeignet ist. Aus dem Modelltyp heraus kann eine passende Modellierungsmethode und Implementation gewählt werden. Ein Simulationsalgorithmus, der auf fast alle Modelltypen angewendet werden kann, ist die Diskrete Ereignisgesteuerte Simulation (DES). Nur zur Modellierung kontinuierlicher Prozesse muß dieser Ansatz erweitert werden. Da aber die Beschreibung des Informationsflusses in einem Fertigungssystem durchwegs auf diskrete Weise erfolgen kann, kann der DES-Ansatz uneingeschränkt angewendet werden.

Im DES-Ansatz wird das zeitliche Verhalten des Systems durch Ereignisse gesteuert, die zu einem Zeitpunkt eintreten und je nach Typ des Ereignisses den Zustand des Systems verändern. Dieser Algorithmus ist in vielfacher Weise, z.B. um eine hierarchische, objektorientierte Modellierung, erweiterbar. Ebenso können verschiedene Modellierungsmethoden auf diesen Algorithmus aufgesetzt werden, die je nach Eigenschaften des Modells und Ziel der Simulation eine geeignete Repräsentation des Modells darstellen.

Aufbauend auf dem Prinzip der diskreten ereignisgesteuerten Simulation existieren eine große Anzahl von Modellierungsmethoden, die jeweils bestimmte Aspekte und Zielsetzungen für Simulationsmodelle unterstützen. Zur Beschreibung eines Fertigungssystems wird eine hierarchische Modellierung benötigt, um die Übersicht über die vielen Objekte des Systems zu behalten. Da die einzelnen Objekte über viele Attribute verfügen, wird ein Konzept zur Darstellung des Objektzustandes benötigt. Dabei besitzen die einzelnen Objekte bei der Modellierung des Informationsflusses im Fertigungssystem kein hochgradig komplexes Verhalten, so daß zur Beschreibung auf dieser Stufe eine einfache Methode ausreichend ist. Im Hinblick auf die potentiellen Anwender muß davon ausgegangen werden, daß diese über keine Programmierkenntnisse verfügen, so daß die Modellierung graphisch und dialogorientiert erfolgen muß. Keine der in Kapitel 3 vorgestellten Modellierungsmethoden kann jedoch alle Anforderungen erfülllen, so daß aus der Kombination verschiedener Ansätze eine eigene Modellierungsmethode entwikkelt wurde.

Das DEVS-Konzept bietet eine hierarchische, objektorientierte Modellierung, die zur Gliederung eines Fertigungssystems übernommen wurde. Dabei wird die Hierarchie aus den in den PPS-Daten vorhandenen Strukturen über Aufträge, Arbeitsplatzgruppen, Einzelarbeitsplätzen usw. gebildet. Allerdings definiert DEVS nicht die Modellierung auf der Stufe der einzelnen Objekte. Ein Ansatz zur Modellierung einzelner Objekte stellen Ereignisgraphen dar, die auf einfache, graphische Weise den Zusammenhang zwischen den einzelnen Ereignissen beschreiben, wobei das Verhalten, das durch die Ereignisse gesteuert wird, in Form von einfachen Regeln dargestellt wird. Dabei wird jedoch der Objektzustand nur implizit reräsentiert. Aus diesem Grund wurde aus der Methode der Petri-Netze das Zustandskonzept übernommen, wobei gleichfalls die graphische Repräsentation dieser Methode zur Darstellung von Zuständen und Ereignissen übernommen wurde. Um dem Anwender eine einfache Beschreibung des Verhaltens zu ermöglichen, wurde das Regelkonzept der Ereignisgraphen um ein mächtiges, dialogorientiertes Regelsystem erweitert, das eine weitgehende Modellierung ohne Programmieraufwand ermöglicht.

8.2. Implementation

Im Rahmen dieser Arbeit wurde die Simulationsshell DYBBS implementiert. Die Implementation erfolgte in Common LISP unter ALLEGRO CL FOR WINDOWS 3.0.2 als zusätzliches Modul zum PPS-Werkzeug WIZARD. Die Implementation berücksichtigt dabei die im vorherigen Abschnitt erläuterten Ansätze und Probleme von Simulations- und PPS-Systemen.

Die Implementation umfaßt drei Teilmodule. Das Statistikmodul enthält grundlegende Funktionen und Klassen zur Zufallszahlenerzeugung und statistischen Auswertung und ist weitgehend eigenständig. Aufbauend auf das Objektsystem von COKE wurde im Simulationskern ein Simulator implementiert, der den in Abschnitt 2.3 beschriebenen DES-Algorithmus beinhaltet. Als Modellierungsmethode wurde aus den in Kapitel 3 erläuterten Methoden eine neue Methode entwickelt, die eine Zerlegung des Fertigungssystems in hierarchisch strukturierbare Elemente gestattet, wobei die Hierarchiebildung anhand der Strukturierung der PPS-Daten erfolgt. Die Modellierung der einzelnen Elemente erfolgt durch einen Zustandsgraph, der sowohl Ereignisse als auch Zustände beinhaltet. Das Verhalten eines Elementes wird durch Regeln beschrieben, wobei in DYBBS die Erstellung der Regeln weitgehend durch eine graphische Benutzeroberfläche erfolgen kann. Durch die Möglichkeiten der Parametrisierung von Regeln können diese leicht wiederverwendet werden.

In einem dritten Integrationsmodul ist die Anbindung der Simulationsshell an WIZARD implementiert, die die vorhandene Funktionalität von WIZARD bei der Erstellung von Simulationsmodellen, bei der Durchführung der Simulation und bei der Analyse der Simulationsergebnisse nutzt. Die Modellerstellung wird durch die vorhandene Datenkopplung vereinfacht, die das Anlegen eines Simulationsmodells mit seiner hierarchischen Gliederung ermöglicht, so daß nur die Modellierung der einzelnen Elemente erfolgen muß. Durch die enge Anbindung an WIZARD ist es möglich, eine Simulation ausgehend von einem Belegungsplan durchzuführen, die Abweichungen zwischen Belegungsplan und Simulationsablauf während der Simulation zu bewerten und bei zu großen Abweichungen während der Simulation Umplanungen durchzuführen. Für die dynamische Bewertung während des Simulationslaufs wurden die in Abschnitt 4.6 aufgezählten Bewertungsmaße wie die Verspätung von Operationen gegenüber dem Plan implementiert, da die traditionellen statischen Kennzahlen nur sehr eingeschränkt zur dynamischen Bewertung während des Ablaufs eingesetzt werden können.

Für die Analyse der Simulationsergebnisse nach Ablauf der Simulation wurden sowohl einfache statistische Auswertungen über den Simulationslauf als auch eine Bewertung der Unterschiede zwischen Belegungsplan und Simulationsablauf implementiert. Bei dieser Ist-Soll-Analyse nach der Durchführung der Simulation kann dabei der komplette Belegungsplan gegen den Simulationsablauf verglichen werden. Als Bewertungskriterien bieten sich hierbei die in Abschnitt 4.4 erläuterten statischen Bewertungskriterien wie die Analyse der Durchlaufzeit an. Sowohl die dynamischen als auch die statischen Bewertungskriterien können in DYBBS leicht um neue Bewertungen ergänzt werden, da die Bewertungen stark vom Produktionsumfeld und den Zielen des PPS-Systems abhängig sind.

8.3. Ergebnisse

Zur Evaluation der implementierten Simulationsshell wurden mehrere kleinere Simulationsmodelle erstellt, die jeweils einen Aspekt von DYBBS aufzeigen. Ausgehend von PPS-Daten einer virtuellen Fertigungsumgebung und ohne die Durchführung einer Systemanalyse zur Gewinnung zusätzlich benötigten Wissens sind die Simulationsergebnisse dieser Modelle nicht auf reale Systeme übertragbar. Die durchgeführten Simulationen zeigen jedoch, für welche Fragestellungen DYBBS eingesetzt werden, wie die Modellierung dazu erfolgen und welche Ergebnisse das Simulationssystem liefern kann. Das Haupteinsatzgebiet von DyBBS wird als Unterstützungswerkzeug zur Disposition gesehen. Daher behandeln die Evaluationsbeispiele Fragestellungen, die sich bei der Disposition ergeben und die mit herkömmlichen Methoden nicht direkt untersucht werden können. Als Beispiele wurden folgende Simulationen durchgeführt:

- **Scheduling durch Simulation:** Erstellen eines Belegungsplanes durch Prioritätsregeln in den Warteschlangen der Ressourcen
- **Zusammenwirken von Werkern und Maschinen:** Zusätzlich zu den Arbeitsplätzen treten hier eine begrenzte Anzahl von Werkern als Engpaß auf. Für einen gegebenen Belegungsplan wird untersucht, wieviele Werker zur Bearbeitung benötigt werden.

- **Einplanung von Eilaufträgen:** Bei der Abarbeitung eines gegebenen Belegungsplanes kommen neue Aufträge hinzu, die während der Abarbeitung eintreffen, bearbeitet werden.
- **Sicherheitszuschläge für die Planung:** In einer störanfälligen Fertigungsumgebung können die Belegungspläne nicht eingehalten werden, weshalb zur Kompensation der Störungen im Plan Sicherheitszuschläge eingebaut werden sollen. Die Simulation liefert Hinweise auf die Größenordnung der zu verwendenden Sicherheiten.

Es konnte gezeigt werden, daß die Modellierung eines Fertigungssystems in DYBBS schnell[21] und zielgerichtet erfolgen kann. Durch die hierarchische Modellierung und einer graphischen Modellierungsmethode, die sowohl ereignis- als auch regelorientiert vorgeht, bleibt das Simulationsmodell übersichtlich und kann leicht wiederverwendet werden.

Die Ergebnisse, die durch Simulation geliefert wurden, konnten mit Hilfe der implementierten Statistik- und Bewertungsroutinen soweit aufbereitet werden, daß sowohl Aussagen über zuvor gesteckte Ziele (z.B. Termintreue) als auch über die statische Qualität der Simulationsergebnisse möglich sind. Allerdings mußten für den Vergleich von Experimenten mit unterschiedlichen Parametern (z.B. Anzahl der Werker oder Sicherheitszuschlag) die Ergebnisse der Einzelexperimente manuell zusammengefasst und ausgewertet werden. An dieser Stelle gibt es noch Möglichkeiten zur Verbesserung.

8.4. Bewertung

Im Rahmen dieser Arbeit wurde versucht, die in der theoretischen Untersuchung der Problemlösungsmethode Simulation gewonnenen Erkenntnisse auf das Einsatzgebiet von PPS-Systemen anzuwenden. Dabei stellte sich heraus, daß grundlegende Methoden der Simulation wie der DES-Algorithmus sehr gut zur Simulation eines Fertigungssystems geeignet sind. Allerdings konnte keine der untersuchten Modellierungsmethoden ohne Nachteile übernommen werden. Daher wurden aus vorhandenen Modellierungsmethoden jeweils bestimmte Elemente zu einer neuen Modellierungsmethode, den Zustandsgraphen, vereinigt. Dieser Ansatz bietet den Vorteil, daß Modelle für die gegebene Problemstellung übersichtlich, schnell und intuitiv erstellt werden können. Dem gegenüber steht der Nachteil, daß eine eigene Modellierungsmethode vom Anwender gelernt werden muß und daß sie die Kommunikation über Simulationsmodelle zwischen den Anwendern verschiedener Simulationssysteme erschwert. Da DYBBS als Unterstützungswerkzeug zur Disposition konzipiert ist, ist vom typischen Anwender zwar viel Erfahrung mit PPS-Werkzeugen, jedoch nur geringe Kenntnisse von Simulation zu erwarten. Aus diesem Grund muß er sich für die Benutzung sowieso in die Thematik einarbeiten, weshalb die Entscheidung für eine eigene Modellierungsmethode vertretbar ist.

Bei der Untersuchung von Bewertungskriterien für Ressourcenbelegungspläne mußte festgestellt werden, daß im Gegensatz zu den über lange Zeit verwendeten statischen Kennzahlen nur sehr wenige Vorschläge zur Bewertung von dynamischen Eigenschaften von Belegungsplänen existieren. Daher wurde in der Implementation Wert darauf gelegt, daß neben einer beispielhaften Grundlage von dynamischen Bewertungsfunktionen eine gute Erweiterbarkeit für neue Bewertungen gegeben ist. Da sowohl die statischen als auch die dynamischen Bewertungen im Anwendungsfall stark vom vorhandenen Produktionssystem und den Unternehmenszielen abhängen, können an dieser Stelle auch keine allgemeingültigen Funktionen geschaffen werden.

Im Gegensatz zu den meisten bestehenden Simulationssystemen bietet DYBBS eine enge Integration mit einem PPS-System. Durch diese Integration eignet sich DYBBS sehr gut zur Modellierung spezieller Probleme innerhalb des Produktionsprozesses mit einer Verknüpfung zu jeweils aktuellen PPS-Daten. Diese Eignung konnte in verschiedenen Beispielsimulationen demonstriert werden. Als Nachteil steht dem ein höherer Zeitaufwand für die Simulation im Vergleich zu kommerziellen Simulatoren gegenüber. Da aber der Zeitaufwand in erträglichen Grenzen gehalten werden konnte, ist dieser Nachteil als weniger gravierend anzusehen.

[21] Für die Modelle der Evaluationsbeispiele wurde pro Modell ca. ein bis zwei Tage benötigt.

8.5. Ausblick

In diesem Abschnitt werden einige Erweiterungen vorgestellt, die z.Zt. in DYBBS fehlen. Ein Teil dieser Erweiterungen kann sehr einfach hinzugefügt werden und wurde nur aus Zeitgründen nicht implementiert, während andere Vorschläge größere Änderungen an der Architektur bedeuten würden.

8.5.1. Statistikmodul

Im Statistikmodul fehlen einige in der Praxis vorkommende Verteilungsfunktionen, insbesondere die Gamma- und Betaverteilungsfunktion. Ebenso sollten einfache statistische Testverfahren, v.a. im Hinblick auf die Testverfahren für mehrere Simulationsläufe oder Experimente mit unterschiedlichen Parametern, eingebaut werden. Für die Verteilungsfunktionen müßten die entsprechenden Algorithmen programmiert werden, während eine ausreichende Anzahl statistischer Testverfahren durch die Einbindung des CLASP[22]-Pakets erfolgen kann.

8.5.2. Simulationsmodul

Durch eine Umstellung des COKE-Objektsystems auf CLOS kann eine Beschleunigung der Simulation sowie eine bessere Strukturierung und Erweiterbarkeit des Quelltextes erreicht werden. Dieser Vorschlag würde aber erhebliche Änderungen in COKE und WIZARD nach sich ziehen. Ebenfalls benötigt wird ein benutzerfreundliches Verhalten bei Fehlern, da es bei bestimmten Fehlern im Modell zu Laufzeitfehlern kommt, die den Benutzer auf die Ebene der LISP-Fehlermeldungen bringen. Eine dritte Erweiterung stellt die Möglichkeit der parallelen Simulation dar, die das gleichzeitige Ausführen mehrerer Simulationsmodelle wie z.B. in SIMPLE++ erlaubt, was allerdings schwierig zu verwirklichen ist.

Die Modellierung kann v.a. im Aspekt der Hierarchisierung erweitert werden. Da es in WIZARD z.Zt. nur die Ebenen der Gruppen- und Einzelressourcen gibt, ist diese Beschränkung auch in DYBBS vorhanden. Hier sollte die Möglichkeit zu einer weitergehenden bzw. anders strukturierten Hierarchiebildung gegeben sein.

8.5.3. Integration in WIZARD

Bei der Integration in WIZARD gibt es Erweiterungsmöglichkeiten, was das Nebeneinander von Belegungsplan und Simulationsablauf angeht. Dieses betrifft zum einen die Visualisierung. Hierbei kann die in WIZARD vorhandene Darstellung einer Belegung so erweitert werden, daß sie gleichzeitig die beiden Belegungen darstellt, so daß die Unterschiede und Gemeinsamkeiten zwischen beiden verdeutlicht werden. Abbildung 27 zeigt einen Vorschlag, wie diese Darstellung aussehen kann. Dabei werden Belegungsplan und Simulationsablauf untereinander abgebildet und die jeweilig gleichen Operationen miteinander verbunden. Zwei ähnliche Belegungspläne sind dann an einer großen Anzahl parallel zueinander verlaufenden Verbindung zu erkennen, während viele Überschneidungen auf Diskrepanzen zwischen beiden Belegungen hinweisen. Aufgrund der in WIZARD vorhandenen Möglichkeiten kann in dieser Anzeige durch Vergrößern die Aufmerksamkeit auf einen bestimmten Bereich gelenkt werden.

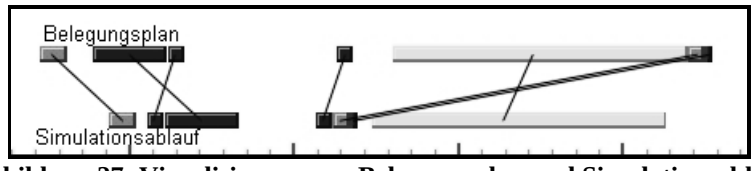


Abbildung 27: Visualisierung von Belegungsplan und Simulationsablauf

[22] CLASP = Common LISP Analytical Statistics Package (ftp://ftp.cs.umass.edu/pub/eksl/clasp)

Auf der anderen Seite sind Erweiterungen bei der Bewertung von Plan und Ablauf denkbar. Insbesondere mit Hinblick auf den in [Sch98] vorgestellten Ansatz eines eigenen Bewertungswerkzeugs für verschiedene Handlungsalternativen ergeben sich hierfür neue Möglichkeiten.

9. Verzeichnisse

9.1. Literaturverzeichnis

[Bau96] BAUSE, F.: *Queueing Petri Nets: A Formalism for the Combined Qualitative and Quantitative Analysis of Systems*, Dortmund, 1996
(http://ls4-www.informatik.uni-dortmund.de/~bause/PN/QPN_article/qpn_final/qpn_final.html)

[Ber97] BERLEANT, D. & KUIPERS, B. J.: *Qualitative and quantitative simulation: bridging the gap*, In: Artificial Intelligence 95, pp. 215-255, 1997

[Bus96] BUSS, A. H.: *Modeling with Event Graphs.* In: Proceedings of the 1996 Winter Simulation Conference, p. 153-160, 1996

[Dux95] DUX, F.: *Entwicklung allgemeiner Konzepte zur Simulation fertigungswirtschaftlicher Probleme: Anwendung der Konzepte auf eine Aktenordnerfertigung*, Diplomarbeit im Fach Informatik am Lehrstuhl für Operations Research und Systemtheorie der Universität Passau, Passau, 1995
(http://www.fmi.uni-passau.de/~angela/dux/Diplomarbeit.Buch_1.html)

[Fer94] FERBER, J.: *Simulating with reactive agents.* In: Many-agent Simulation and Artificial Life, Hillebrand, E. & Stender J. (Eds.), pp. 8-31, Amsterdam, 1994, IOS Press

[Fis92] FISHWICK, P. A., ZEIGLER, B. P.: *A Multimodel Methodology for Qualitative Model Engineering.* In: ACM Transactions on Modeling and Computer Simulation, Vol 2, No. 1, pp. 52-81,1992

[Gus94] GUSTAVSON, A. & TÖRN, A.: *XsimNet, A Tool in C++ For Executing Simulation Nets.* In: Proceedings of the Conference on Modelling and Simulation 1994, p. 146-150, 1994

[Hac89] HACKSTEIN, R.: *Produktionsplanung und -steuerung (PPS): ein Handbuch für die Betriebspraxis*, 2. Auflage, Düsseldorf, 1989, VDI-Verlag

[Hes95] HESTERMANN, C. & POECK, K.: *Intelligentes, kooperatives Assistenzsystem zur Dispositionsunterstützung in Produktionsplanungs- und -steuerungssystemen (INKAD).* In: KLAUCK, C. & MÜLLER, J. (HRSG.): Künstliche Intelligenz und verteilte PPS-Systeme; Beiträge des 1. Bremer KI-Pfingstworkshops, Universität Bremen, FB Informatik, Juni 1995

[Hes97] HESTERMANN, C.: *COKE und WIZARD, Technische Referenz*, Würzburg, 1997, erhältlich auf Anfrage beim Autor
(mailto:hester@informatik.uni-wuerzburg.de)

[Hes97-2] HESTERMANN, C.: *Wissensbasierte Any-Time Suche bei diskreter und kontinuierlicher Zuordnung*, eingereicht zur XPS-97, Universität Würzburg, 1997, erhältlich auf Anfrage beim Autor
(mailto:hester@informatik.uni-wuerzburg.de)

[Hie96] HIERL, J.: *WIMFRIED - Entwurf und Implementierung eines wissensbasierten Systems zur integrierten Mittelfristdisposition*, Diplomarbeit am Lehrstuhl für Künstliche Intelligenz und Angewandte Informatik, Universität Würzburg, Würzburg, 1996

[Hoo89] HOOVER, S. V. & PERRY, R. F.: *Simulation: a problem solving approach*, Reading (MA), 1989

[Jen92-1] JENSEN, K.: *Coloured Petri Nets - Volume 1: Basic Concepts*, Berlin, 1992, Springer

[Jen92-2] JENSEN, K.: *Coloured Petri Nets - Volume 2: Analysis Methods*, Berlin, 1992, Springer

[Jos96] JOS, R. S. & MIYAGI, P. E.: *A Formal Approach To PFS/MFG: A Petri Net Representation of Discrete Manufacturing Systems*. In: Studies in Informatics and Control, Vol. 5, No. 2, 1996

[Ker94] KERNLER, H.: *PPS der 3. Generation: Grundlagen, Methoden, Anregungen*, 2. Auflage, Heidelberg, 1994, Hüthig

[Klu97] KLUEGL, F.: *Multiagentensysteme und Simulation, Seminar im WS96/97 am Lehrstuhl Informatik VI, Universität Würzburg*, Würzburg, 1997, erhältlich auf Anfrage beim Autor
(mailto:kluegl@informatik.uni-wuerzburg.de)

[Kur93] KURBEL, K.: *Produktionsplanung und -steuerung: Methodische Grundlagen von PPS-Systemen und Erweiterungen*, München, 1993, Oldenbourg

[Law91] LAW, A. M. & KELTON D. W.: *Simulation Modeling And Analysis*, 2nd Edition, New York, 1991, McGraw-Hill

[Mos93] MOSER, M.: *Regelung der Maschinenbelegung in der flexiblen Fertigung*, Düsseldorf, 1993, VDI-Verlag

[Pag94] PAGE, E. H. JR.: *Simulation Modeling Methodology: Principles And Etiology Of Decision Support*, Blacksburg (VA), 1994

[Par92] PAREDIS, J. & VAN RIJ, T.*: Simulation and Constraint Programming as Support Methodologies for Job Shop Scheduling.* In: Journal Of Decision Systems (Revue des systèmes de décision), Vol. 1, No. 1, 1992, p. 59-77, Paris, 1992, Hermès

[Poe95] POECK, K.: *Konfigurierbare Problemlösungsmethoden am Beispiel der Problemklassen Zuordnung*

und Diagnostik, St. Augustin, 1995, infix

[Pup90] PUPPE, F.: *Problemlösungsmethoden in Expertensystemen*, Berlin, 1990, Springer

[Ros83] ROSENSTENGEL, B. & WINAND, U.: *Petri-Netze: eine anwendungsorientierte Einführung*, 2. Auflage, Braunschweig, 1983, Vieweg

[Sau95] SAUER W., WEIGERT G. & GOERIGK P.: *Real time Optimization of Manufacturing Processes by Synchronized Simulation.* In: 5th International Conference on Flexible Automation and Intelligent Manufacturing, S. 271-282, Stuttgart, 1995, Begell house

[Sau96] SAUER W., WEIGERT G. & GOERIGK P.: *Simulationsgestützte optimale Steuerung flexibler Fertigungssysteme.* In: Proceedings of 7th International DAAAM Symposium 17-19th October 1996, S. 389-390, Wien, 1996

[Sch98] SCHMIDT, L.: *Verbundprojekt: Unterstützung der Entscheidungsfindung in der mittelfristigen Produktionsplanung*, Fachgebiet Wirtschaftsinformatik I, Institut für Wirtschaftsinformatik, TU Illmenau, 1998

[Sch87] SCHMIDT, R.: *Einsatzmöglichkeiten der Simulation in der Werkstattsteuerung*, In: Simulationstechnik - Proceedings des 4. Symposiums Simulationstechnik Zürich, Halin, J. (Hrsg.), p. 521-538, Berlin u.a., 1987, Springer

[Ste90] STEELE, G. L.: *Common LISP - The Language, 2nd Edition*, 1990, Digital Press

[Tou93] TOUSSAINT, A.: *Modellierung eines „Standardwerkers": Einbindung der Arbeiter in die Simulation zur Personalbedarfs- und Ablaufplanung.* In: Simulation In Passau, 1/1993, S. 14-17, Passau, 1993

[Tra96] TRAN-GIA, P.: *Analytische Leistungsbewertung verteilter Systeme: Eine Einführung*, Berlin, 1996, Springer

[Vea94] VEACH M. S., CODDINGTON P. & FOX G. C.: *BURN: A Simulation of Forest Fire Propagation.*In: Journal of Undergraduate Research in High-Performance Computing, Volume 4, E.A. Bogucz (ed.), Northeast Parallel Architectures Center at Syracuse University Technical Report, Syracuse, 1994
(http://www.npac.syr.edu/REU/reu94/abstracts.html#veach)

[Wei96] WEIMAR, J.R.: *Simulation with Cellular Automata: Lecture Notes*, Braunschweig, 1996
(http://www.tu-bs.de/institute/WiR/weimar/Zascript/)

[Wit94] WITTE, T., CLAUS, T. & HELLING, K.: *Simulation von Produktionssystemen mit SLAM; Eine praxisorientierte Einführung*, 1994

[Zei93] ZEIGLER, B. P. & SANKAIT V.: *DEVS Formalism and Methodology: Unity of Conception / Diversity of Application*, In: Proceedings of the 1993 Winter Simulation Conference, pp. 573-579, Los Angeles, 1993

[Zel92] ZELL, M.: *Simulationsgestützte Fertigungssteuerung*, München, 1992, Oldenbourg

9.2. Abbildungsverzeichnis

ABBILDUNG 1: VERFAHREN DES KRITISCHEN EREIGNISSES (AUS [LAW91], S. 12) 13
ABBILDUNG 2: WARTESCHLANGE MIT BEDIENEINHEIT (AUS [LAW91], S. 46) 20
ABBILDUNG 3: EVENT GRAPH FOR TRANSFER LINE MODEL (AUS [BUS96], S. 158) 20
ABBILDUNG 4: EINFACHES PETRI-NETZ 25
ABBILDUNG 5: AUFLÖSUNG DES VERZWEIGUNGSKONFLIKTS (AUS [ROS83], S. 28, ABB. 18) 27
ABBILDUNG 6: ACTIVITY CYCLE DIAGRAM FÜR DAS ENGLISH PUB MODEL (AUS [PAG94], S. 52) 32
ABBILDUNG 7: KONTROLLFLUßGRAPH FÜR BEDIENEINHEIT MIT UNTERBRECHUNG (AUS [PAG94], S. 72F) 33
ABBILDUNG 8: JOHN CONVAY'S „GAME OF LIFE" (AUS [WEI96], FIG. 1) 35
ABBILDUNG 9: ERZEUGNISSTRUKTUREN ALS GOZINTO-GRAPH (AUS [KUR93], S. 64, ABB. 2.2-3) 41
ABBILDUNG 10: VORSCHLAGEN-UND-VERTAUSCHEN-STRATEGIE (AUS [HES95], S. 5) 45
ABBILDUNG 11: DIAGRAMME ZUR DARSTELLUNG VON TERMINTREUE, DLZ, LAGERBESTAND UND AUSLASTUNG (NACH [KER94], S. 226, ABB. 7.16-7.19) 50
ABBILDUNG 12: VERGLEICH ZWISCHEN PLAN UND ABLAUF 56
ABBILDUNG 13: BEISPIEL AUS DER ARENA-MODELLBIBLIOTHEK 58
ABBILDUNG 14: ARENA, SERVEREIGENSCHAFTEN 59
ABBILDUNG 15: AUFTRAGSGESTEUERTE FERTIGUNG IN ARENA 60
ABBILDUNG 16: XROSI - BEISPIELMODELL 62
ABBILDUNG 17: SIMULATIONSVERLAUF IN XROSI 63
ABBILDUNG 18: KLASSENDEFINITION IN COKE 65
ABBILDUNG 19: ÜBERSICHT ÜBER DIE BEZEICHNUNGEN IN DYBBS 67
ABBILDUNG 20: SYSTEMARCHITEKTUR IN DYBBS 70
ABBILDUNG 21: ZUSTANDSGRAPH 74

ABBILDUNG 22: EVALUATION - M/M/1-WARTESYSTEM 80
ABBILDUNG 23: AUSLASTUNG FÜR SZENARIO 4 82
ABBILDUNG 24: ERGEBNISSE DES ARBEITSPLATZ-PERSONAL-BEISPIELS 84
ABBILDUNG 25: EILAUFTRÄGE EINPLANEN - PASS ERROR 86
ABBILDUNG 26: EVALUATION - PARAMETERLAUF 87
ABBILDUNG 27: VISUALISIERUNG VON BELEGUNGSPLAN UND SIMULATIONSABLAUF 93
ABBILDUNG 28: HANDBUCH - NACHFOLGERGRAPH 104
ABBILDUNG 29: HANDBUCH - ZUSTANDSGRAPH 105
ABBILDUNG 30: HANDBUCH - ELEMENTEIGENSCHAFTEN 106
ABBILDUNG 31: HANDBUCH - MODELLPARAMETER 107
ABBILDUNG 32: HANDBUCH - EREIGNISSE EDITIEREN 108
ABBILDUNG 33: HANDBUCH - REGEL EDITIEREN 109
ABBILDUNG 34: HANDBUCH - ARGUMENTDEFINITION FÜR EINE REGEL 110
ABBILDUNG 35: HANDBUCH - VORBEDINGUNGEN EDITIEREN 111
ABBILDUNG 36: HANDBUCH - AUSWIRKUNGEN (LISP-CODE) 113
ABBILDUNG 37: HANDBUCH - AUSWIRKUNGEN (EREIGNISDEFINITION) 114
ABBILDUNG 38: HANDBUCH - SIMULATIONSOPTIONEN 116
ABBILDUNG 39: HANDBUCH - ANIMATION 117
ABBILDUNG 40: HANDBUCH - ERGEBNISTABELLE 117
ABBILDUNG 41: HANDBUCH - SOLL-IST-VERGLEICH (DIAGRAMM) 118
ABBILDUNG 42: HANDBUCH - AUSLASTUNGSTABELLE 119

9.3. Verzeichnis der WWW-Adressen

Produkt	Hersteller	URL
Allegro CL for Windows	Franz Inc.	http://www.franz.com/
ARENA	Systems Modeling Inc.	http://www.sm.com/
SiMPLE++	Tecnomatix	http://www.tecnomatix.com/
XROSI	Institut für Elektonik-Technologie der TU Dresden	http://www.et.tu-dresden.de/iet/ietproz-rosi.html
MS Visual Basic	Microsoft Gmbh	http://www.microsoft.com/de/vb/
MS Excel	Microsoft Gmbh	http://www.microsoft.com/office/
Macintosh Common LISP (MCL)	Digitool, Inc.	http://www.digitool.com/
VPPS	infor Gmbh	http://www.infor.de/
Inkad-Projekt	u.a. Lehrstuhl Informatik VI, Universität Würzburg	http://www-info6.informatik.uni-wuerzburg.de/~inkad/
CLASP	University of Massachusetts	ftp://ftp.cs.umass.edu/pub/eksl/clasp

ANHANG

Anhang A: Modellbibliothek

In diesem Abschnitt werden die wichtigsten in DYBBS implementierten Modelle beschrieben. Für jedes Modell wird eine kurze Beschreibung und eine Erläuterung der Modellparameter gegeben. Die vorgestellten Modelle sind im Datenverzeichnis von DYBBS abgelegt und können als Vorlagen für eigene Simulationsmodelle verwendet werden.

Quelle

Name	Quelle-MM1	
Beschreibung	erzeugt in exponentiell verteilten Abständen Dummy-Operationen. Diese Quelle kann zur Modellierung von Warteschlangennetzen ohne Berücksichtigung von PPS-Daten verwendet werden	
Modellparameter	Ankunftsrate	durchschnittliche Dauer zw. 2 Anforderungen
Verwendung	Abschnitt 7.1	

Name	Auftragsquelle
Beschreibung	gibt die ersten Operationen der im Modell vorhandenen Aufträge gemäß ihrer FT frei
Modellparameter	-

Name	Operationenquelle
Beschreibung	gibt alle Operationen, die Modell vorhanden sind, gemäß ihrer FT frei. Die Arbeitsplätze müssen entscheiden, ob die Operationen ausgeführt werden können (d.h. ob die Vorgänger beendet sind).
Modellparameter	-

Name	Nachfolger freigeben
Beschreibung	gibt zu Beginn der Simulation alle Operationen ohne Vorgänger frei. Erhält die Quelle während der Simulation über das Ereignis „Ankunft“ eine beendete Operation, so gibt sie den Nachfolger zu dessen FT frei. Für dieses Modell müssen die Arbeitsplätze als Nachfolger die Quelle eingetragen haben.
Modellparameter	-

Arbeitsplatz

Name	M-M-1	
Beschreibung	modelliert eine Bedienstation der Kapazität 1 mit exponentiell verteilten Bedienzeiten ohne Berücksichtigung von PPS-Daten. Erweiterungen dieses Modells sind die Modelle M-M-1-failure, das eine zufällige Ausfallwahrscheinlichkeit zu Beginn der Bearbeitung und einer festen Reparaturdauer anbietet, und M-M-n, das eine Bedienstation der Kapazität n bereitstellt.	
Modellparameter	Bedienrate	mittlere Bediendauer
Verwendung	Abschnitt 7.1	

Name	Arbeitsplatz-einfach	
Beschreibung	Grundmodell für Arbeitsplätze mit PPS-Daten. Die Arbeitsplätze haben die Kapazität 1 und fallen nicht aus. In bearbeiteten Operationen werden in den entsprechenden Datenstrukturen Eintragungen über den Starttermin und die Dauer gemacht. Dieses Modell kann für das Scheduling auf Einzelarbeitsplätzen verwendet werden. Das Modell 'Arbeitsplatz - Gruppe einfach' ist analog, nur daß es auch auf Arbeitsplatzgruppen verwendet werden kann.	
Modellparameter	Bearbeitungsdauer	Dauer der aktuell bearbeiteten Operation
Verwendung	Abschnitt 7.2	

Name	Arbeitsplatz mit Störung	
Beschreibung	Modell eines Einzelarbeitsplatzes, der zu zufälligen Zeitpunkten ausfallen kann. Ausfälle können nur während einer Bearbeitung auftreten und haben eine zufällige Dauer.	
Modellparameter	Bearbeitungsdauer	Dauer der aktuell bearbeiteten Operation

	Reparaturdauer	Dauer des letzten oder aktuellen Ausfalls
	Ausfallverteilung	Verteilungsfunktion für den Zeitraum zwischen Reparaturende und nächstem Ausfall
	Reparaturdauerverteilung	Verteilungsfunktion für die Dauer der Reparatur

Name	Arbeitsplatz mit Störung - Gruppe	
Beschreibung	wie Arbeitsplatz mit Störung, nur daß auch Gruppenarbeitsplätze mit einer Kapazität > 1 berücksichtigt werden können. Der Ausfall betrifft alle Einzelarbeitsplätze gleichzeitig, d.h. für die Störungsdauer steht keine Kapazität zur Verfügung.	
Modellparameter	wie Arbeitsplatz mit Störung	
	akt-parallel	Anzahl der aktuell genutzten Kapazität
Verwendung	Abschnitt 7.4, 7.5	

Name	Arbeitsplatz mit Personal	
Beschreibung	Einzelarbeitsplatz ohne Ausfall. Zum Durchführen der Bearbeitung wird ein Werker benötigt, der das Ereignis „Ankunft“ über den möglichen Start der Bearbeitung erhält und das Ende der Bearbeitung über das Ereignis „Fertig“ an den Arbeitsplatz signalisiert.	
Modellparameter	Bearbeitungsdauer	Dauer der aktuell bearbeiteten Operation
	Schichtgruppen	Personalgruppen-Elemente, die diejenigen Werker enthalten, die den Arbeitsplatz bedienen können.
Verwendung	Abschnitt 7.3	

Name	Arbeitsplatz - Gruppe verteilt auf Einzeln
Beschreibung	Modell einer Arbeitsplatzgruppe, der die an die Gruppe gerichteten Operationen auf einen Einzelarbeitsplatz der Gruppe weiterleitet.
Modellparameter	-
Verwendung	Abschnitt 7.3

Name	Arbeitsplatz - Reparatur durch Techniker	
Beschreibung	wie Arbeitsplatz mit Störung - Gruppe, nur daß die Reparatur von außen durch einen Servicetechniker vollzogen wird. Dabei kann die Reparatur durch beschränkte Kapazitäten der Techniker verzögert werden. Der Techniker muß über die Ereignisse „Reparatur“ und „Fertig“ Beginn und Ende der Reparatur signalisieren. Dieses Modell kann auch auf Einzelarbeitsplätze angewendet werden.	
Modellparameter	wie Arbeitsplatz mit Störung - Gruppe	
	Technikergruppe	Personalgruppe, die für die Reparatur des Arbeitsplatzes zuständig sind.

Personal

Name	Personal-einfach	
Beschreibung	wie Arbeitsplatz-einfach, ebenso können fast alle Arbeitsplatzmodelle auch für Personalelemente verwendet werden	
Modellparameter	Bearbeitungsdauer	Dauer der aktuell bearbeiteten Operation

Name	Schichtgruppe
Beschreibung	verteilt ankommende Anforderungen an eine Personalgruppe auf einen einzelnen Werker
Modellparameter	-
Verwendung	Abschnitt 7.3

Name	Schicht
Beschreibung	Modell für einzelne Werker, die nur während ihrer Schichtzeit zur Verfügung stehen. Am Schichtende werden alle wartenden Anforderungen an die Nachfolgerschicht übergeben.
Modellparameter	-
Verwendung	Abschnitt 7.3

Name	Servicetechniker	
Beschreibung	repariert ausgefallene Arbeitsplätze. Dieses Modell ist das Gegenstück zum Modell Arbeitsplatz - Reparatur durch Techniker. Ein Techniker kann immer nur einen Arbeitsplatz reparieren, weitere ausgefallene Arbeitsplätze müssen warten.	
Modellparameter	Reparaturdauer	Dauer der aktuellen Reparatur
	Reparaturdauerverteilung	Verteilungsfunktion zur Reparaturdauer

Operation

Name	Operation-einfach
Beschreibung	einfaches Modell für Operationen. Dieses Modell enthält keine Regeln und dient dazu, die Operationen in die entsprechende Zustände zu versetzen. Die dazu notwendigen Ereignisse müssen vom Arbeitsplatz der Operation kommen.
Modellparameter	-

Auftrag

Name	Auftrag-einfach
Beschreibung	einfaches Modell für Aufträge. Dieses Modell enthält keine Regeln und dient dazu, die Aufträge in die entsprechende Zustände zu versetzen.
Modellparameter	-

Bewerter

Name	Planbewerter	
Beschreibung	einfaches Modell für das Element Planbewerter. Zu regelmäßigen Zeitpunkten oder auf ein Signal hin wird eine Bewertung des Simulationsablaufes durchgeführt und bei Überschreitung der Grenzwerte für die jeweilige Bewertungsfunktion eine Umplanung durchgeführt. Grundlage der Bewertung sind alle im Simulationsmodell vorhandenen Operationen.	
Modellparameter	Minimaldauer	Mindestdauer zwischen zwei Bewertungen
	Maximaldauer	Maximaldauer zwischen zwei Bewertungen
	Schranken	abhängig von der jeweiligen Bewertungsfunktion des Planbewerterelements.

Senke

Name	Senke-einfach
Beschreibung	einfaches Modell für das Element Senke, das keine Regeln beinhaltet. In vielen Modellen wird nach Beendigung der letzten Operation das Ereignis „Ankunft“ an die Senke geschickt, das daraufhin z.B. einen Auftrag für beendet erklären kann.
Modellparameter	-

Name	Senke-MM1
Beschreibung	Modell für Warteschlangennetze ohne PPS-Daten. In diesen Modellen werden Dummy-Operationen erzeugt, die in der Senke wieder vernichtet werden.
Modellparameter	-
Verwendung	Abschnitt 7.1

Anhang B: Benutzerhandbuch

In diesem Kapitel wird die Benutzung von DYBBS erläutert. Der grundlegende Aufbau und Anwendungsgebiete der Simulationsshell DYBBS sind in Kapitel 6 und 7 dargelegt. Das folgende Handbuch ist eine Zusammenfassung des Online-Handbuchs zu DYBBS, das während der Benutzung kontextsensitiv zur Verfügung steht.

Konventionen

Folgende Textformatierungen bedeuten:

`Button`	Verweis auf Dialogelemente wie Menüpunkte, Eingabefelder, Buttons usw.
`eingabe`	Verweis auf eine Tastatureingabe
Definition	Einführung eines Begriffes

Bezeichnungen

Ausgehend von den in Abschnitt 6.2 eingeführten Bezeichungen werden alle Objekte der PPS-Daten (Arbeitsplätze, Personal, Aufträge, Operationen, usw.) und alle simulationsspezifischen Objekte (Warteschlange, Simulator) als ***Simulationselemente*** oder kurz ***Elemente*** bezeichnet. Die jeweilige Klasse eines Elements wird auch ***Elementklasse*** genannt. Für jedes Element kann das Verhalten des Elements in der Simulation definiert werden. Diese Spezifikation für ein Element wird als ***Modell*** bezeichnet. Ein Modell besteht aus einem ***Zustandsgraphen,*** der die Zustandsübergänge des Elements steuert, der Festlegung der ***Elementeigenschaften*** und den ***Modellparametern***, die zusätzliche Daten über das Modell enthalten. Während Elementeigenschaften von der Elementklasse abhängen und fest vorgegeben sind, können Modellparameter vom Benutzer frei definiert werden und bestehen aus einer Menge von Variable-Wert-Paaren. Die Menge der Modelle über alle Elemente wird als ***Simulationsmodell*** bezeichnet. Zusätzlich kann auch für eine Elementklasse ein Modell erstellt werden. Dieses wird als ***Klassenmodell*** bezeichnet und kann als Vorlage für einzelne Objekte dieser Elementklasse verwendet werden.

Aufbau

Die folgenden Punkte erläutern die Modellierung, wobei das Vorgehen in einem "top-down"-Ansatz beschrieben wird:

1. Nachfolgergraph
2. Zustandsgraph
3. Eigenschaften des Modells festlegen
4. Zustände definieren
5. Ereignistypen definieren
6. Zustände editieren
7. Ereignisse editieren
8. Definition von Regeln
9. Regeln hinzufügen
10. Definition von Vorbedingungen
11. Definition von Auswirkungen
12. Modell überprüfen

Die weiteren Punkte betreffen die Organisation des Programms und den Ablauf der Simulation:

13. Laden und Speichern
14. Simulationsoptionen

15. Systemoptionen
16. Simulationslauf
17. Experimentkontrolle
18. Simulationsergebnisse

1. Nachfolgergraph

Im Nachfolgergraph können Elemente in das Simulationsmodell aufgenommen, entfernt und die Beziehungen der Elemente untereinander festgelegt werden. Die unterschiedlichen Elementklassen werden durch verschiedene Icons dargestellt, die im Menü `Systemoptionen` geändert werden können.

Durch Anklicken eines Elementes mit der linken Maustaste erhält man den Zustandsgraph für dieses Element, mit der rechten Maustaste die Eigenschaften des Elements. Durch einen Mausklick mit der rechten Maustaste auf den Hintergrund erhält man ein Menü. Alle diese Aktionen können ebenso über die Menüleiste angewählt werden.

Über `neues Element` kann wie in der COKE-Objektdefinition ein neues Objekt angelegt werden. Allerdings wird ein so erzeugtes Objekt nicht automatisch dem Simulationsmodell hinzugefügt. Dazu muß es über `Elemente hinzufügen` zusätzlich in die Simulationsumgebung übernommen werden. Über `Elemente löschen` kann es wieder aus der Simulationsumgebung entfernt werden. Dabei wird allerdings das Objekt selbst nicht gelöscht. Dies kann über die COKE-Objektdefinition erfolgen. Bei `Elemente editieren` können ein oder mehrere Elemente ausgewählt werden, die bearbeitet werden. Wenn mehrere Elemente ausgewählt werden, so kann für diese Elemente gleichzeitig nach Auswahl eines Referenzelementes ein Zustandsgraph definiert werden.

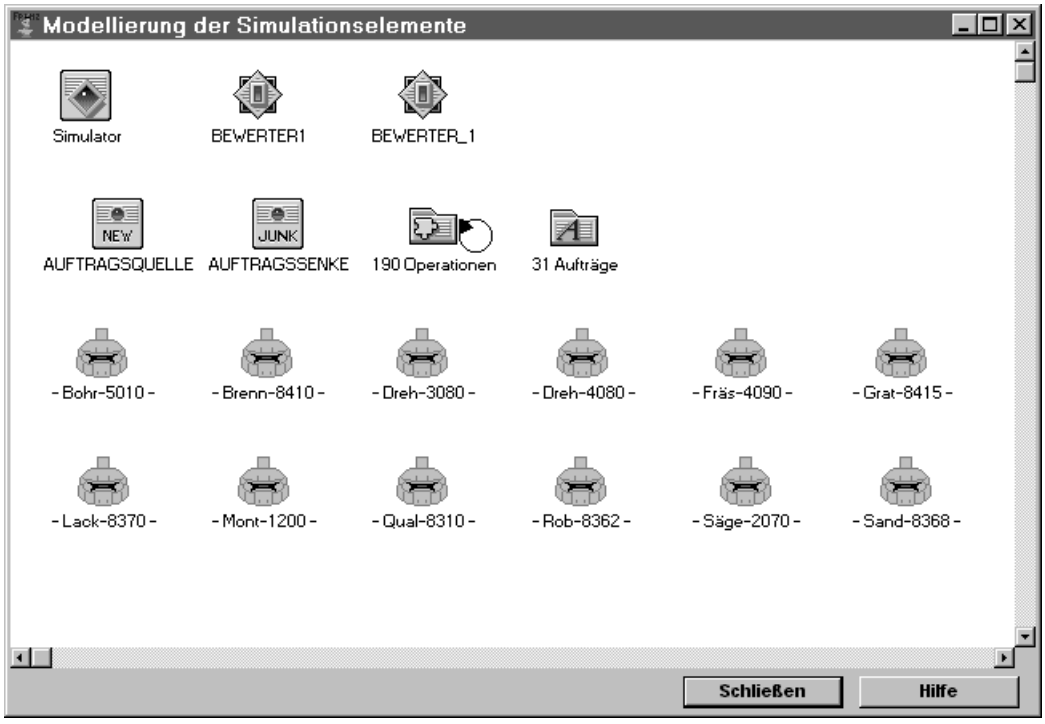


Abbildung 28: Handbuch - Nachfolgergraph

Über `Modell laden` kann ein zuvor gespeichertes Simulationsmodell geladen werden. Dabei wird das bestehende Simulationsmodell überschrieben. Elemente, die noch nicht existieren, aber im gespeicherten Simulationsmodell vorhanden sind, werden nachträglich angelegt. Die Ressourcen und Fertigungsstrukturen müssen allerdings zuvor durch die Kopplung in WIZARD geladen werden, da nur die simulationsspezifischen, nicht aber die PPS-spezischen Daten gespeichert werden. Beim Speichern eines Simulationsmodells werden die Zustandsgraphen aller im Simulationsmodell vorhandener Elemente sowie deren Eigenschaften und Modellparameter, nicht aber die Elemente selbst gespeichert (s. Anhang B.13).

2. Zustandsgraph

Im Zustandsgraph wird das Verhalten eines Elementes festgelegt. Diese Definition kann allgemein für eine Elementklasse oder speziell für einzelne Elemente durchgeführt werden. Im Zustandsgraph wird das Verhalten eines Elementes durch seine Zustandsübergänge, die durch Ereignisse ausgelöst werden, dargestellt.

Zustände werden durch Ovale, Ereignisse durch Rechtecke dargestellt. Diese Konvention wurde aus dem Konzept der Petri-Netzen (s. Abschnitt 3.3) übernommen. Der Ausgangszustand ist durch eine dicke Umrandung gekennzeichnet. Wenn für das Ereignis eine Ereigniswahrscheinlichkeit (ungleich 1) eingegeben wurde, wird diese über dem Ereignis wiedergegeben. Alle Regeln, die mit einem Ereignis verbunden sind, sind unter dem Ereignis aufgezählt. Die Farben für Zustände und Ereignisse können im Menü `Systemoptionen` geändert werden.

Zustände können in drei Gruppen eingeteilt werden, die für die Animation und die Ergebnisauswertung eine spezielle Rolle spielen:

- **Arbeitszustände** fassen alle Zustände zusammen, in der das Element beschäftigt ist.
- **Ausfallzustände** sind diejenigen Zustände, in denen das Element nicht arbeiten kann, obwohl Operationen zur Bearbeitung vorhanden sind.
- Alle anderen Zustände sind Zustände, in denen das Element weder arbeitet noch gestört ist.

Diese Zustandsgruppen werden durch drei verschiedene Farben (Default: Rot für gestört, Grün für arbeitend, Gelb für alle anderen) angezeigt.

Eine Kante von einem Zustand über ein Ereignis zu einem Zustand bedeutet, daß das Element durch das Ereignis in den neuen Zustand übergehen kann, wenn eine mit dem Ereignis verbundene Regel feuert oder keine Regel angegeben ist. Von einem Zustandsknoten können mehrere Kanten zu Ereignisknoten führen, von einem Ereignisknoten darf aber nur eine Kante zu einem Zustandsknoten verlaufen.

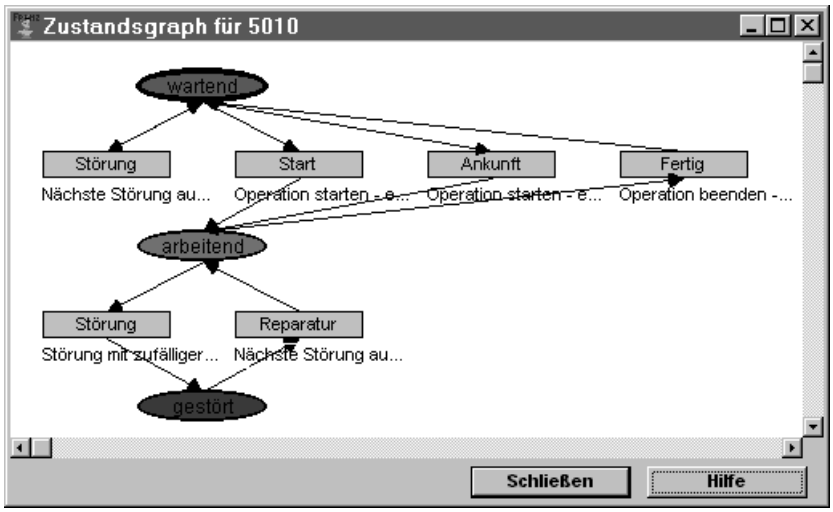


Abbildung 29: Handbuch - Zustandsgraph

Über das Kontextmenü können verschiedene Aktionen ausgewählt werden. Befindet sich der Mauszeiger auf einem Zustand oder Ereignis, so erscheinen Menüs, die die jeweiligen Möglichkeiten anzeigen. Durch Anklicken der Zustände oder Ereignisse können diese editiert werden. Diese Aktionen werden in den folgenden Abschnitten erläutert.

3. Eigenschaften von Simulationselementen

Elementeigenschaften

Für jedes in das Simulationsmodell hinzugefügte Element können bestimmte Eigenschaften festgelegt werden. Andere Eigenschaften, die nicht in DYBBS benötigt werden, können in der Objekteingabe von COKE verändert werden. Je nachdem, zu welcher Klasse das Element gehört, sind bestimmte Einstellungen möglich. Wenn mehrere Elemente gleichzeitig editiert werden, können nur bestimmte Einstellungen in diesem Dialog festgelegt werden.

`Name` und `Beschreibung` identifizieren das Element. Der Name muß eindeutig sein und sollte nur aus alphanumerischen Zeichen bestehen. Weiterhin läßt sich einschalten, ob für das Element die Statistik- und Animationsroutinen aktiviert werden sollen. Besitzt das Element eine Warteschlange (alle Ressourcenklassen), so kann daraufhin die Warteschlangendisziplin und ein Test auf die Verfügbarkeit des wartenden Elements ausgewählt werden. Ob die Warteschlange blockiert, wenn das erste wartende Objekt nicht verfügbar ist, kann ebenfalls eingestellt werden. Eine blockierende Warteschlange erlaubt nicht das Überholen innerhalb der Warteschlange, wenn das erste wartende Objekt nicht verfügbar ist.

Der Nachfolger eines Elements dient zur Festlegung des Informationsflusses im System und kann entweder statisch oder dynamisch bestimmt werden. Beispielsweise kann sich der Nachfolger eines Arbeitsplatzes aus der aktuell bearbeiteten Operation ergeben. Soll der Nachfolger dynamisch zur Laufzeit berechnet werden, so kann entweder eine Berechnungsformel als LISP-Code oder eine bereits vorhandene Funktion ausgewählt werden.

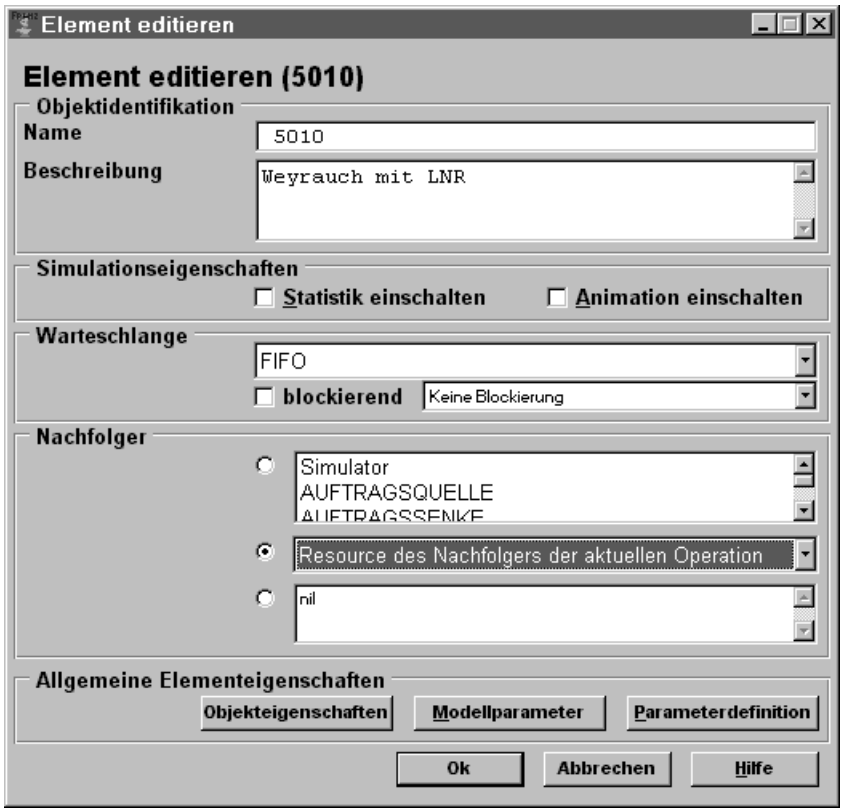


Abbildung 30: Handbuch - Elementeigenschaften

Modellparameter

Modellspezifische Parameter werden zusammen mit dem Zustandsgraph gespeichert und enthalten Attribute, die nur für das Modell eine besondere, vom Benutzer festzulegende Bedeutung haben. Wenn ein Modell zu einem Element geladen wurde, können diese Parameter wie Klassenattribute von COKE verwendet werden. Allerdings können für mehrere Elemente der gleichen Klasse (z.B. zwei verschiedene Arbeitsplätze) zwei unterschiedliche Modelle mit unterschiedlichen Modellparametern geladen werden.

Jedes Element besitzt dann die für sein Modell definierten Modellparameter. In diesem Sinne sind die Modellparameter instanzenspezifische Attribute von Objekten im Gegensatz zu Klassenattributen. Die Eingabe der modellspezifischen Parameter erfolgt analog zur COKE-Attributdefinition für Klassen. Zuerst muß über `Parameterdefinition` ein Parameter angelegt werden (s. Abbildung 31). Zur Definition werden ein Name und ein Typ benötigt, eine Beschreibung kann zusätzlich eingegeben werden.

Abbildung 31: Handbuch - Modellparameter

Die für ein Element definierten Modellparameter können über `Modellparameter` mit Werten belegt werden. Während der Simulation können die Werte durch entsprechende Auswirkungen geändert werden. Nach dem Simulationslauf werden die Defaultwerte wiederhergestellt. Die Eingabe der Werte erfolgt analog zur Objekteingabe in COKE. Je nach Typ des Parameters erfolgt die Eingabe als Zahl, Text, Termin, Dauer usw.

Auf die Modellparameter kann in Regeln, Vorbedingungen und Auswirkungen durch ihren Namen verwiesen werden. Sie verhalten sich, als wären sie lokale Variablen. Allerdings können die Variablen nur ausgelesen werden, zum Verändern der Attribute sollte eine Auswirkung vom Typ Attributsveränderung verwendet werden. Es sollten keine Namenskollisionen zwischen Attributen, Regel-, Vorbedingungs- und Auswirkungsnamen vorliegen, da solche Kollisionen können zu Programmfehlern führen.

4. Zustände definieren

Zustände geben, an in welcher Situation sich ein Element zum aktuellen Simulationszeitpunkt befindet und welche Zustandsänderungen von diesem Zustand ausgehend eintreten können. Um einen Zustand in einen Zustandsgraphen einzubauen, muß er vorher definiert werden. Ein neuer Zustand kann im Menü `Zustände` definiert werden. Dazu muß der Name des Zustandes eingegeben werden. Im weiteren Verlauf kann der Zustand immer über seinen eindeutigen Namen referenziert werden. Im gleichen Menü können auch Zustände gelöscht werden, wenn sie nicht mehr verwendet werden sollen. Dabei wird beim Löschen nicht die Verwendung des Zustandes im Zustandsgraphen gelöscht. Diese müssen manuell vom Benutzer entfernt werden (s. auch Anhang B.12).

5. Ereignistypen definieren

Ein Ereignis ermöglicht den Zustandswechsel eines Elementes. Wenn ein Ereignis an einem Element eintrifft, so wird geprüft, ob für den aktuellen Zustand des Elements und den Ereignistyp des eingetretenen Elements ein Übergang im Zustandsgraph exisitiert. Wie bei Zuständen muß der entsprechende Ereignistyp zuvor definiert werden. Ein neuer Ereignistyp kann im Menü `Ereignistyp` definiert werden, indem der Name des Ereignistypen eingegeben wird. Analog zu Zuständen werden Ereignistypen gelöscht.

6. Zustände editieren

Um einen Zustand in einen Zustandsgraphen einzubauen, muß er vorher definiert worden sein (s. Anhang B.4). Danach kann er in einem Zustandsgraph durch Anwahl von `Neuer Zustand` ausgewählt und hinzugefügt werden. Ebenso kann ein Zustand gelöscht werden. Das Löschen löscht den Zustand und

alle damit verbundenen Ereignisse aus dem Zustandsgraphen. Bestimmte Zustände sind bereits vordefiniert und sollten nicht gelöscht werden:

Elementklasse	Zustände	Kommentar
Simulator	`wartend, arbeitend, angehalten`	systemintern
Warteschlange	`wartend`	systemintern

Tabelle 7: Handbuch - Vordefinierte Zustände

Ein Zustand ist als Startzustand besonders ausgezeichnet. Der Startzustand ist derjenige Zustand, in der sich das Element zu Beginn der Simulation befindet. Er kann nicht gelöscht werden. Daher muß gegebenenfalls vorher ein neuer Startzustand ausgewählt werden. Sinnvollerweise sollten zuerst alle Zustände in den Zustandsgraph eingebaut werden, bevor die Übergänge zwischen den Zuständen durch Ereignisse hergestellt werden.

Abbildung 32: Handbuch - Ereignisse editieren

7. Ereignisse editieren

Für ein Ereignis muß ein Typ ausgewählt werden, der zuvor definiert werden muss (s. Anhang B.5). Danach werden einer oder mehrere Ausgangszustände festgelegt. Die Ausgangszustände müssen vorher in den Zustandsgraph aufgenommen worden sein. Der Folgezustand ist derjenige Zustand, in dem sich das Element nach dem Ereignis befindet, wenn der Zustandsübergang erfolgt ist. Dies kann auch wiederum einer der Ausgangszustände sein. Wenn bereits ein Ereignis mit dem gleichen Ereignistyp, Ausgangs- und Folgezustand existiert, wird es überschrieben. Es können aber Ereignisse definiert werden, die den gleichen Typ und den gleichen Ausgangszustand, aber unterschiedliche Folgezustände besitzen. In diesem Fall müssen die Regeln so spezifiziert werden, daß nur eine Regel feuert. Ansonsten wird eine Regel ausgeführt und die anderen ignoriert, ohne daß spezifiziert ist, welche Regel feuert. Ob der Zustandswechsel erfolgt, hängt weiterhin von der Ereigniswahrscheinlichkeit ab (Zahl zw. 0 und 1 oder ein Modellparameter), die angibt, mit welcher Wahrscheinlichkeit das Ereignis überhaupt am Element ankommt.

Folgende Ereignisse sind durch das System vordefiniert. Sie dürfen nicht gelöscht werden und können direkt in eigenen Modellen verwendet werden:

Elementklas	Ereignis	Kommentar
Simulator	Start, Stop, Simulation unterbrechen, Simulation fortsetzen	systemintern
Warteschlange, Arbeitsplatz, Personal, e-Material, Werkzeug	Ankunft	Die Weitergabe an den Nachfolger löst Element mit Warteschlange bei der War das Ereignis Ankunft aus. Der Inhalt des E nisses wird in die Warteschlange eing erhält das Element selbst das Ereigni
Quelle	Ankunft	Alle Quellen erhalten beim Start der Ereignis Ankunft
alle Elemente	Stop, simulation starten	beim Initialisieren bzw. Beenden der

Tabelle 8: Handbuch - Vordefinierte Ereignisse

8. Definition von Regeln

Welche Veränderungen außer dem Zustandsübergang, der bereits im Zustandsgraphen festgelegt ist, ein Element aufgrund eines Ereignisses durchläuft, wird in DYBBS mit Hilfe von Regeln spezifiziert. Eine Regel besteht aus zwei Bestandteilen: In der Vorbedingung steht eine Prädikatsfunktion. Ist die Vorbedingung erfüllt, dann werden die Auswirkungen der Regel ausgeführt.

Zur Definition einer Regel müssen zuerst ein eindeutiger `Name` und eine `Beschreibung` eingegeben werden.

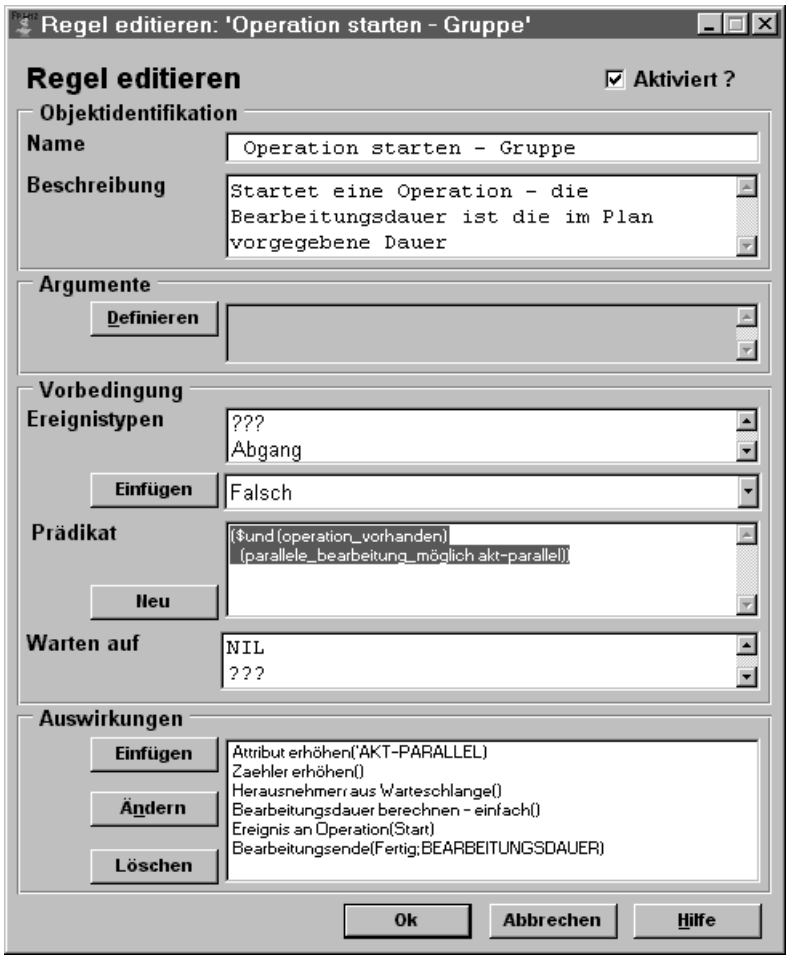


Abbildung 33: Handbuch - Regel editieren

Die `Argumente` der Regel können wie in der Attributdefinition von COKE eingegeben werden. Argumente dienen zur Parametrisierung von Regeln. Die an die Regel übergebenen Argumente können wie Argumente an eine Prozedur aufgefaßt werden. Insbesondere können sie an die Vorbedingung und Auswirkungen der Regel weitergereicht werden. Bei der Definition der Argumente wird jeweils ein Name

und ein Wertetyp benötigt (s. Abbildung 34). Wird eine Regel in einem Zustandsgraphen verwendet, so können für die Argumente der Regel Konstanten oder Modellparameter eingesetzt werden. Analog zu den Parametern für Regeln können Auswirkungen und Vorbedingungen (s.u.) parametrisiert werden.

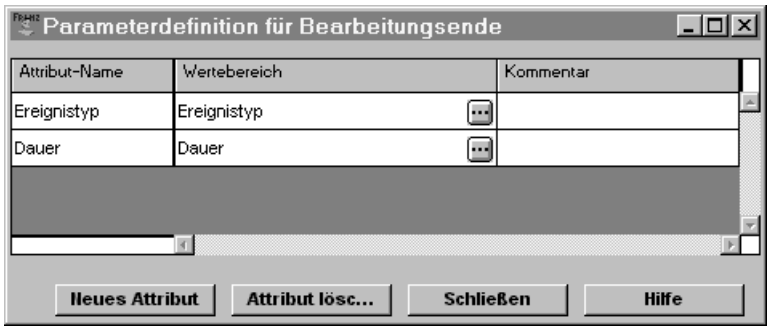


Abbildung 34: Handbuch - Argumentdefinition für eine Regel

In den weiteren Dialogfeldern des Regeldialogs kann ein Prädikat für die Regel erstellt werden. Es können mehrere Ereignistypen definiert werden, auf welche die Anwendung der Regel sinnvoll ist. Eine Einschränkung der Ereignistypen erlaubt eine bessere Strukturierung bei der Ereignisdefinition. Wenn ein Ereignis an einem Element eintrifft, und die Regel dazu ist nicht für den Typ des Ereignisses vorgesehen, so wird die Regel ignoriert.

Bei der Angabe der Prädikatsfunktion ist eine Verknüpfung aus einzelnen Vorbedingungen möglich. Über `Einfügen` kann eine in der Auswahlbox ausgewählte Vorbedingung in das Prädikat eingebaut werden. Wenn die Vorbedingung Argumente erfordert, müssen in einem zusätzlichen Dialog Werte für die Argumente eingegeben werden. Als Werte für die Argumente sind Konstanten und Argumentnamen der Regelargumente zulässig. Soll auf Modellparameter eines Elementes zugegriffen werden, so sollten diese als Argumente an die Regel übergeben werden. Die Syntax des Prädikates ist ein LISP-Ausdruck. Daher ist darauf zu achten, daß sowohl die Klammerung als auch die Aufrufe der einzelnen Vorbedingungen syntaktisch korrekt ist. Die einzelnen Vorbedingungen können durch beliebige LISP-Ausdrücke miteinander verknüpft werden. U.a. können die Makros der COKE-Kommandosprache hier verwendet werden, z.B. `$=, $ODER, $UND`. Wie Vorbedingungen definiert werden, ist bei der Definition von Vorbedingungen (s. Anhang B.10) beschrieben.

Wenn das Prädikat nicht erfüllt ist, wird das Ereignis im Normalfall ignoriert und gelöscht. Da es vorkommen kann, da man das gleiche Ereignis später wieder aufrufen will, wenn sich Veränderungen ergeben haben, so daß das Prädikat nun erfüllt sein könnte, können ein oder mehrere Ereignistypen angegeben werden, auf die gewartet werden soll. Wenn bei einer Regel diese angegeben werden, so wird das Ereignis nicht gelöscht, sondern wartet darauf, daß ein Ereignis von den angegebenen Typen an diesem Element eintrifft. Ist dies der Fall, wird das Ereignis erneut aktiviert und sein Prädikat wird erneut geprüft.

Schließlich müssen die Auswirkungen angegeben werden, die ausgeführt werden, wenn das Prädikat erfüllt ist. Die Auswirkungen werden dabei in der Reihenfolge ausgeführt, in der sie aufgelistet sind. Über die Buttons `Einfügen` und `Löschen` können die auszuführenden Auswirkungen in der Liste verändert werden. Wird vor dem Einfügen eine vorhandene Auswirkung der Liste angewählt, so wird vor dieser Auswirkung eingefügt. Beim Einfügen einer Auswirkung können in einem weiteren Dialog die Auswirkung sowie deren Parameter eingeben werden.

Zum Löschen von Regeln können im Menü `Regeln löschen` eine oder mehrere Regeln ausgewählt werden. Das Löschen einer Regel löscht zwar die Regel, nicht aber ihre Verwendung in den Zustandsgraphen. Dort müssen die gelöschten Regeln manuell aus den Ereignissen entfernt werden (s. auch Anhang B.12). Weiterhin können Regeln deaktiviert werden. Wenn eine Regel nicht aktiviert ist, wird sie während des Simulationslaufes nicht ausgeführt. Die Aktivierung von Regeln erfolgt unter dem Menüpunkt `Regeln aktivieren`

9. Regeln hinzufügen

Nachdem Regeln erstellt worden sind, können sie in Ereignisse eingebaut werden. Beim Editieren von Ereignissen (s. Anhang B.7) können über die Buttons `Einfügen`, `Ändern` und `Löschen` die für dieses Ereignis zuständigen Regeln eingebaut werden. Es können mehrere Regeln angegeben werden, wobei solange Regeln in der Reihenfolge ihrer Auflistung ausprobiert werden, bis eine feuert. Es wird also nur eine Regel ausgeführt, es können aber mehrere Alternativen angegeben werden.

Über den Button `Einfügen` erhält man eine Auswahl aus denjenigen Regeln, die für den Ereignistyp des Ereignisses zugelassen wurden. Dazu müssen zusätzlich die Parameter der Regeln festgelegt werden. Als Parameter können außer konstanten Werten auch Modellparameter (Anhang B.3) angegeben werden.

10. Definition von Vorbedingungen

Eine Vorbedingung ist ein Bestandteil der linken Hälfte von Regeln, die angibt, wann die Regel feuert. Die linke Hälfte ist ein logischer Ausdruck, der bei seiner Auswertung „wahr“ oder „falsch“ liefert. In DYBBS kann dieser logische Ausdruck aus einzelnen Vorbedingungen zusammengesetzt sein, die über logische Verknüpfungen miteinander in Beziehung gesetzt werden. In der Sprechweise der Logik sind Vorbedingungen die „Atome“ eines logischen Ausdrucks. Wie die Vorbedingungen zu einem Ausdruck zusammengesetzt werden, ist bei der Definition von Regeln (s. Anhang B.8) beschrieben.

Abbildung 35: Handbuch - Vorbedingungen editieren

Eine Vorbedingung besteht zuerst aus einem `Namen` und einer `Beschreibung`. Die `zugehörige Klasse` gibt an, auf welche COKE-Klasse sich die Vorbedingung bezieht. Diese Angabe ist nützlich zur Strukturierung. Die `Argumente` werden wie bei Regeln angegeben. `Lokale Variablen` können verwendet werden, wenn zuerst aus den Argumenten noch Zwischenwerte berechnet werden müssen. Die lokalen Variablen werden wie bei einem `let`-Ausdruck in LISP sequentiell ausgewertet. Als geklammerter Ausdruck werden jeweils paarweise Name und Wert der Variablen beschrieben. So bedeutet

```
((a 1)
 (b (+ a 3))
```

daß `a` den Wert `1` und `b` den Wert `a+3` bekommt. Für die Wertzuweisungen können beliebige LISP-Ausdrücke eingesetzt werden.

Die `Formel` selbst ist ein LISP-Ausdruck. In ihr kann auf die Parameter und die lokalen Variablen zugegriffen werden. Desweiteren stehen folgende Variablen zur Verfügung:

`$selbst`	das gerade aktive Element
`$ereignis`	das Ereignis, das den Übergang ausgelöst hat
`$zeitpunkt`	der Zeitpunkt, an dem sich die Simulation befindet
`$zustand`	der aktuelle Zustand des Elements

Diese Variablen sollten allerdings nur gelesen, nicht beschrieben werden. Das Ergebnis der Bedingungsfunktion bestimmt den Wert der Vorbedingung. Liefert der Ausdruck `NIL`, so ist die Vorbedingung nicht erfüllt, jeder andere Wert wird als wahr interpretiert.

Zum `Löschen` von Vorbedingungen müssen im entsprechenden Menüpunkt eine oder mehrere Vorbedingungen ausgewählt werden. Das Löschen einer Vorbedingung erfolgt analog zum Löschen von Regeln. Wenn eine Vorbedingung nicht aktiviert ist, wird während des Simulationslaufes nicht die Gültigkeit der Vorbedingung geprüft. Eine inaktive Vorbedingung liefert immer „falsch“. Die Aktivierung von Vorbedingungen erfolgt unter dem Menüpunkt `Vorbedingungen aktivieren`

11. Definition von Auswirkungen

Eine Auswirkung ist ein Bestandteil der rechten Hälfte einer Regel, der angibt, welche Aktionen durch das Feuern einer Regel ausgelöst werden. Die rechte Hälfte besteht aus einer geordneten Menge solcher Aktionen. In DyBBS sind insgesamt vier verschiedene Typen von Auswirkungen implementiert, deren Erstellung im folgenden beschrieben werden. Wie die Auswirkungen in Regeln eingebaut werden können, ist bei der Definition von Regeln (s. Anhang B.8) beschrieben.

Zur Zeit sind in DyBBS folgende Typen implementiert:

- **LISP-Code**: in diesen können Auswirkungen durch LISP-Ausrücke programmiert werden.
- **Attributsveränderung**: Auswirkungen dieser Art lassen die Veränderung von Elementattributen zu.
- **Ereignisgenerierung:** diese Auswirkungen erzeugen Ereignisse.
- **Umplanung:** Damit kann das Verhalten des Planers gesteuert werden.

Das Löschen und Aktivieren von Auswirkungen erfolgt wie bei Regeln und Vorbedingungen.

Gemeinsame Eigenschaften

Eine Auswirkung benötigt einen `Namen` und kann eine `Beschreibung` haben. Die `zugehörige Klasse` gibt an, auf welche COKE-Klasse sich die Auswirkung bezieht. Die `Argumente` werden wie bei Regeln angegeben. Der Button `Verwendung` listet alle Regeln auf, in der die Auswirkung aufgerufen wird.

LISP-Code

LISP-Code-Auswirkungen verlangen vom Benutzer, daß er die Auswirkung in LISP-Programmcode selbst verfasst. Diese Art der Auswirkung läßt dem Anwender Freiheiten, wofür er allerdings Kenntnisse über LISP, COKE und den Aufbau von DyBBS besitzen muß. `Lokale Variablen` können wie in Vorbedingungen verwendet werden.

Die `Formel` selbst ist ebenfalls ein LISP-Ausdruck. In ihr kann auf die Argumente der Auswirkung und die lokalen Variablen zugegriffen werden. Desweiteren stehen die in den Vorbedingungen erläuterten Variablen zur Verfügung.

Abbildung 36: Handbuch - Auswirkungen (LISP-Code)

Attributsveränderung

Bei der Attributsveränderung können sowohl Modellparameter als auch Klassenattribute von Elementen verändert werden. Die Modellparameter werden nach einem Simulationslauf auf ihren Ursprungswert zurückgesetzt, Klassenattribute dagegen nicht. Daher sollte eine Veränderung von Klassenattributen nur sehr sparsam und wohlüberlegt eingesetzt werden. Anstatt direkt auf den Klassenattributen zu arbeiten, sollten diese im Zweifelsfall in einen neuen Modellparameter kopiert und dort manipuliert werden (s. Anhang B.3).

Zuerst muß ein Attribut bestimmt werden. Wenn ein Klassenattribut verwendet werden soll, kann dieses aus der Liste der Klassenattribute einer Klasse ausgewählt werden. Für einen Modellparameter muß der Name des Parameters eingegeben werden. Dabei muß bei der Ausführung der Auswirkung der Modellparameter für das Element definiert sein. Soll das Attribut auf einen festen Wert gesetzt werden, so kann dieser über die Eingabe `Wert` eingegeben werden. Die Eingabeweise ist abhängig vom Typ des Attributs und verläuft nach den COKE-Konventionen. Soll für das Attribut ein Wert berechnet werden, so muß eine Formel eingegeben werden. Die Formel ist ein LISP-Ausdruck, der den Wert berechnet. Dabei stehen außer den Argumenten der Auswirkung und den üblichen Werten folgende Ausdrücke zur Verfügung:

`$wert`	der alte Wert des Attributs

Der Ausdruck `(1+ $wert)` bedeutet also, daß der Wert des Attributs um `1` erhöht wird.

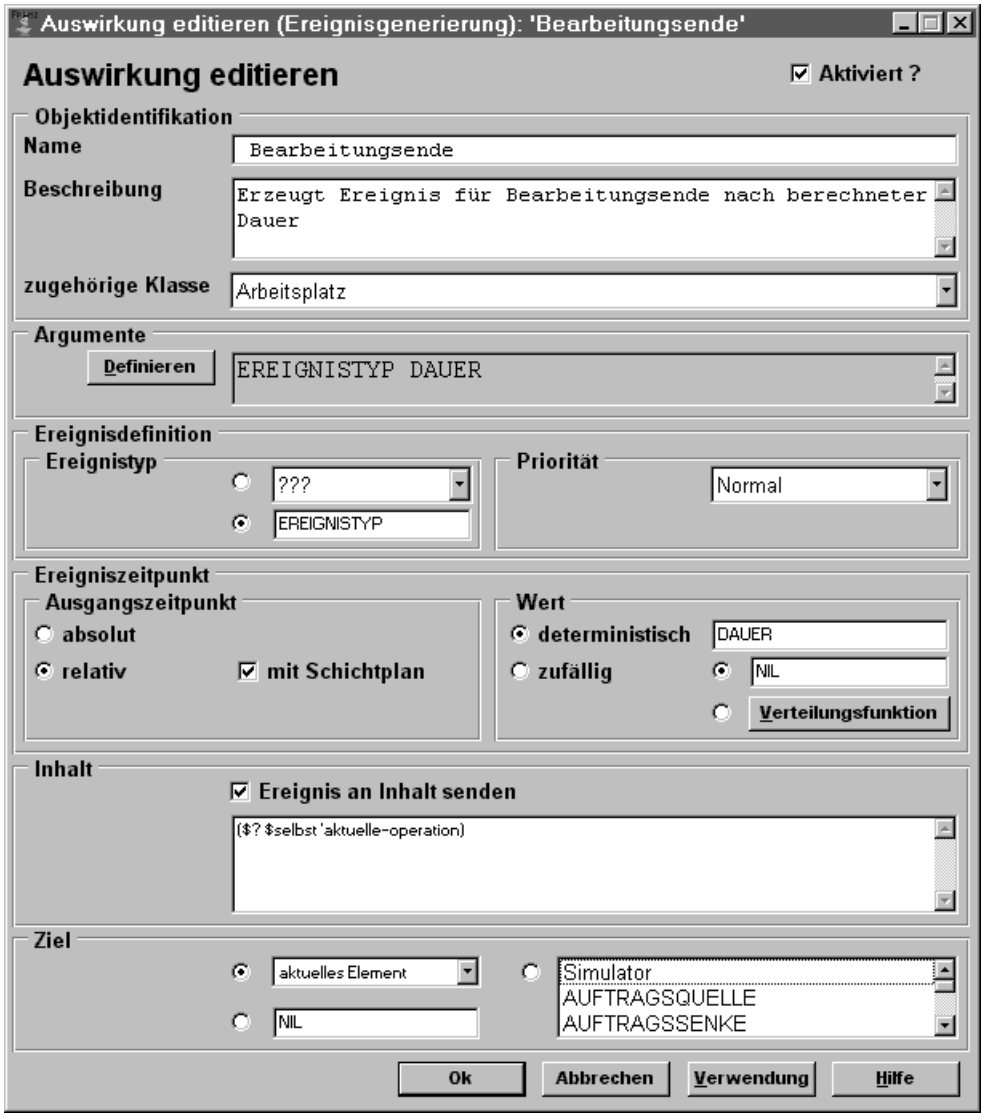


Abbildung 37: Handbuch - Auswirkungen (Ereignisdefinition)

Ereignisgenerierung

Auswirkungen dieses Typs ermöglichen das Erzeugen von Ereignissen. Für ein Ereignis ist hauptsächlich die Angabe eines Ereignistyps und eines Zeitpunktes notwendig. Der Zeitpunkt eines Ereignisses bestimmt sich aus zwei Faktoren. `Relativ` bedeutet, daß die Berechnung zuzüglich des aktuellen Zeitpunktes erfolgt, `absolut` bedeutet, daß allein der angegebene Wert den Zeitpunkt bestimmt. Steht der Zeitpunkt fest, so kann über den Schalter `deterministisch` eine Angabe eines Termins oder Dauer der Zeitpunkt fest eingegeben werden. Erfolgt die Berechnung des Zeitpunktes über eine Zufallsfunktion, so muß der Schalter `zufällig` benutzt werden und über `Verteilungsfunktion` eine Zufallsverteilung bestimmt werden. Zur Angabe der Verteilungsfunktion müssen eine Funktion ausgewählt und ihre Parameter bestimmt werden. Als Parameter können Zahlen und Argumente der Auswirkung eingesetzt werden. Die in DYBBS implementierten Verteilungsfunktionen sind in Anhang C beschrieben.

Ein Ereignis kann einen Inhalt bekommen, der als LISP-Ausdruck eingegeben werden muß. Dabei kann ein beliebiger LISP-Ausdruck eingegeben werden, der zur Laufzeit ausgewertet wird. In der obigen Abbildung ist ein Ausdruck angegeben, der die aktuelle auf dem Arbeitsplatz bearbeitete Operation ermittelt. Ist der Inhalt ein Simulationselement, so kann über den Schalter `Ereignis an Inhalt sende` auch dieses Objekt von dem Ereignis benachrichtigt werden. Im obigen Fall gibt der Ausdruck eine Operation zurück, die ebenfalls ein Ereignis erhält.

Das Ziel des Ereignisses bestimmt, wer von diesem Ereignis betroffen ist. Dabei stehen drei Möglichkeiten zur Verfügung. Das Ziel kann per Auswahl aus einer vorgegebenen Liste von dynamischen Zielen, durch die Eingabe eines LISP-Ausdrucks oder statisch festgelegt werden. In der Regel wird das Ziel dynamisch bestimmt. Dynamische Ziele und LISP-Ausrücke werden zur Laufzeit ausgewertet, statische Ziele sind konstant. Im obigen Beispiel bedeutet das gewählte dynamische Ziel, daß das Element, das ein Ereignis erhalten hat und aufgrundedessen die gezeigte Auswirkung ausführt, sich selbst (und ihrer aktuellen Operation) ein Ereignis schickt.

Umplanung

Um eine Umplanung durchzuführen, kann zuerst der Simulationszustand bewertet werden. Dazu dienen die Instanzen der Klasse **Planbewerter**. Von diesen werden i.d.R. auch die Auswirkungen des Typs Umplanung aufgerufen. Zur Umplanung ist dabei das Planungsszenario und der Umfang der umzuplanenden Elemente zu bestimmen. Der Umfang ist eine Menge von umzupanenden Operationen und kann dabei als Formel, dynamisch oder statisch bestimmt werden.

12. Modell überprüfen

Über den Menüeintrag `Modell überprüfen` kann eine Konsistenzüberprüfung des Simulationsmodells ausgelöst werden. Diese sucht fehlende Ereignistyp- oder Zustandsverwendungen in Zustandsgraphen und überprüft die Argumente von Regeln, Auswirkungen und Vorbedingungen. Dabei wird jedoch nur eine Syntaxprüfung durchgeführt, d.h. es wird auf eine korrekte Verwendeng der Argumente, z.B. im Hinblick auf die Anzahl der Argumente, geachtet. Eine semantische Prüfung findet nicht statt.

13. Laden und Speichern

In DYBBS werden verschiedene Daten in unterschiedlichen Dateien gespeichert:

- Alle Vorbedingungen, Auswirkungen, Regeln und Experimentkontrollen werden in der COKE-Wissensbasis gespeichert.
- Die globalen Simulationsdaten (Zustände, Ereignistypen, Systemoptionen) werden unter dem Menüpunkt `Globale Simulationsdaten speichern` in einer eigenen Datei abgelegt.
- Die einzelnen Elemente (Aufträge, Operationen, Arbeitsplätze, Personal, Werkzeug, Material) werden über die `Kopplung` in das System geladen.
- Modelle für Elementklassen und einzelne Elemente enthalten den Zustandsgraph, Elementeigenschaften und die Modellparameter. Das Speichern und Laden kann im Kontextmenü des jeweiligen Zustandsgraphen angewählt werden.
- In einem Simulationsmodell werden im Menüpunkt `Simulationsmodell speichern` für alle in die Simulation aufgenommenen Elemente die Zustandsgraphen und Modellparameter gespeichert.

Beim Laden sollte folgende Reihenfolge eingehalten werden: Zuerst wird eine Wissensbasis geladen, darauf die globalen Simulationsdaten. Danach sollte die Kopplung mit den PPS-Daten inklusive einem Zuordnungslauf durchgeführt werden. Jetzt kann entweder ein komplettes Simulationsmodell geladen oder mit der Erstellung eines neuen Modells begonnen werden.

14. Simulationsoptionen

In den Simulationsoptionen können verschiedene Einstellungen zum Ablauf der Simulation gemacht werden. `Starttermin` und `Stoptermin` geben an, über welchen Zeitraum die Simulation laufen soll. Wenn ein Simulationsmodell vorliegt, das eine bestimmte Menge von Aufträgen bzw. Operationen simuliert, sollte der Starttermin vor dem FT der ersten Operation und der Stoptermin nach dem SET der letzten Operation liegen. Die `Anzahl der Simulationsläufe` gibt an, wieviele unabhängige Wiederholungen der Simulation gemacht werden. Die Ergebnisse der Läufe werden zusammengefasst. Wenn aus der vorhan-

denen Zuordnung nur eine bestimmte Menge von Operationen oder Aufträge simuliert werden sollen, so kann über den Button `Auswählen` eine Auswahl getroffen werden.

Im `Einzelschrittmodus` wird jedes Ereignis einzeln ausgeführt. Die Option `ausführliches Protokoll` gibt während der Simulation jedes aufgetretene Ereignis aus. Beide Optionen sind zur Fehlersuche gedacht. Das Protokoll kann zusätzlich in eine Datei geschrieben werden. Wenn diese Option ausgeschaltet ist, wird das Simulationsergebnis in einem Bildschirmfenster dargestellt. Desweiteren können Sie die während des Laufes auftretenden Ereignisse von ausgewählten Ereignistypen in eine Datei schreiben bzw. zuvor geschriebene Ereignisse wieder einlesen. In diesem Fall werden diese Ereignisse zur Simulation verwendet und es werden keine neuen Ereignisse erzeugt.

Abbildung 38: Handbuch - Simulationsoptionen

15. Systemoptionen

In den Systemoptionen können verschiedene Einstellungen zur graphischen Benutzeroberfläche und zur Organisation der Simulationsdaten festgelegt werden. Die meisten Einstellungen betreffen die Farben und Icons im Zustands- und Nachfolgergraph. Der Button `Datenverzeichnis` gibt an, in welchem Verzeichnis die Simulationsdaten gespeichert werden. Dies betrifft sowohl die globalen Daten als auch die Modelldaten. Die Systemoptionen werden in den globalen Daten gespeichert.

16. Simulationslauf

Während eines Simulationslaufs können Sie die Simulation anhalten. Daraufhin kann der Simulationsablauf geändert werden. In der `Ereignisagenda` sind alle bekannten, aber noch nicht ausgeführten Ereignisse aufgelistet, die daraufhin manuell verändert werden können.

17. Experimentkontrolle

Ähnlich zur ***Ablaufkontrolle*** in COKE kann eine Experimentkontrolle definiert werden, welche die Definition von Experimenten und den Ablauf der Simulation in einem Skript steuert. Auf diese Weise können Simulationsläufe mit verschiedenen Parameterwerten auf einem Simulationsmodell durchgeführt werden.

18. Simulationsergebnisse

Zur Analyse der Simulationsergebnisse sind in DYBBS vier Anzeigen implementiert. Weiterhin können die Ergebnisse in numerischer Form in eine Textdatei exportiert werden, um mit einem externen Analysewerkzeug ausgewertet zu werden.

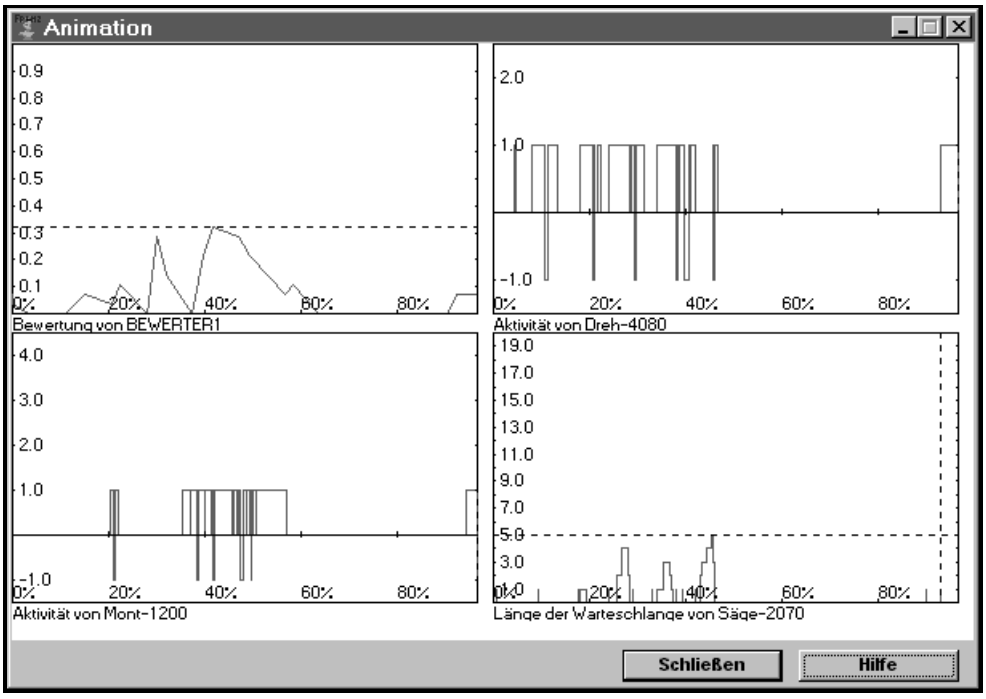
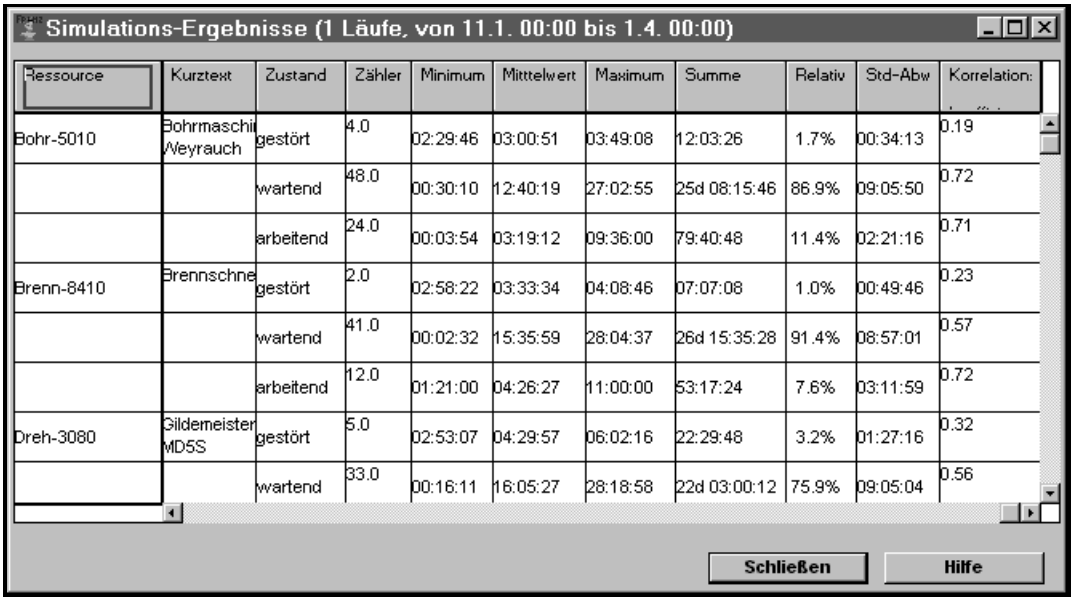


Abbildung 39: Handbuch - Animation

Animation

Die Animationsgraphik stellt einen einfachen graphischen Überblick über die Aktivität einzelner Elemente dar. Bei Ressourcen wird die Aktivität der Ressource, bei Warteschlangen die Länge der Warteschlange und bei Planbewertern das Ergebnis der Bewertung im zeitlichen Verlauf dargestellt. Es erfolgt eine Darstellung für die Zustandsgruppen arbeitend, wartend und gestört. Um die Animation eines Elementes anzuschalten, muß der entsprechende Schalter zur Animation des Elements in den Simulationsoptionen (s. Anhang B.14) gesetzt werden. Die X-Achse der Animationsgraphiken ist immer die Zeitachse, während auf der Y-Achse je nach Element verschiedene Werte aufgetragen sind. Die Aktivität eines Elements bewegt sich zwischen -1 (Störung) und der maximalen Kapazität des Elements. Warteschlangen zeigen den Inhalt der Warteschlange in wartenden Operationen und Bewerter das Ergebnis der Berwertung (zw. 0 und 1) an.



Ressource	Kurztext	Zustand	Zähler	Minimum	Mittelwert	Maximum	Summe	Relativ	Std-Abw	Korrelation:
Bohr-5010	Bohrmaschi Weyrauch	gestört	4.0	02:29:46	03:00:51	03:49:08	12:03:26	1.7%	00:34:13	0.19
		wartend	48.0	00:30:10	12:40:19	27:02:55	25d 08:15:46	86.9%	09:05:50	0.72
		arbeitend	24.0	00:03:54	03:19:12	09:36:00	79:40:48	11.4%	02:21:16	0.71
Brenn-8410	Brennschne	gestört	2.0	02:58:22	03:33:34	04:08:46	07:07:08	1.0%	00:49:46	0.23
		wartend	41.0	00:02:32	15:35:59	28:04:37	26d 15:35:28	91.4%	08:57:01	0.57
		arbeitend	12.0	01:21:00	04:26:27	11:00:00	53:17:24	7.6%	03:11:59	0.72
Dreh-3080	Gildemeister MD5S	gestört	5.0	02:53:07	04:29:57	06:02:16	22:29:48	3.2%	01:27:16	0.32
		wartend	33.0	00:16:11	16:05:27	28:18:58	22d 03:00:12	75.9%	09:05:04	0.56

Abbildung 40: Handbuch - Ergebnistabelle

Ergebnistabelle

Die Ergebnistabelle errechnet eine einfache statistische Auswertung. Dabei werden Statistiken über die Verweildauern von Elementen in den einzelnen Zuständen berechnet. Für jedes Element und jeden Zustand wird dabei Anzahl, Minimum, Mittelwert, Maximum und Gesamtzeit berechnet, in der sich das Element in diesem Zustand gewesen ist. Bei mehreren Läufen erfolgt die Mittelung über alle Läufe. Die gleichen Ergebnisse können auch in einem Balkendiagramm dargestellt werden, wobei für jeden Zustand die Mittelwerte dargestellt werden.

Soll-Ist-Vergleich

Im Soll-Ist-Vergleich können die Simulationsergebnisse mit dem Belegungsplan verglichen werden. Dazu muß zu Beginn ein Vergleichskriterium aus der Liste der vorhandenen Kriterien ausgewählt werden. In DYBBS sind z.Z. folgende Vergleichskriterien implementiert:

- Bearbeitungszeit, DLZ, Wartezeit
- Starttermin, Endtermin
- Puffer vorne, hinten, gesamt
- Auslastung

Daraufhin werden für die Elemente die Daten des Vergleichs berechnet und in einer Tabelle dargestellt. Je nach Kriterium erfolgt der Vergleich auf Ressourcen- oder Auftragsebene. Dargestellt wird die kleinste, durchschnittliche und größte Abweichung zwischen Simulationslauf und Plan. Die Durchschnittswerte können zusätzlich in einem Balkendiagramm dargestellt werden.

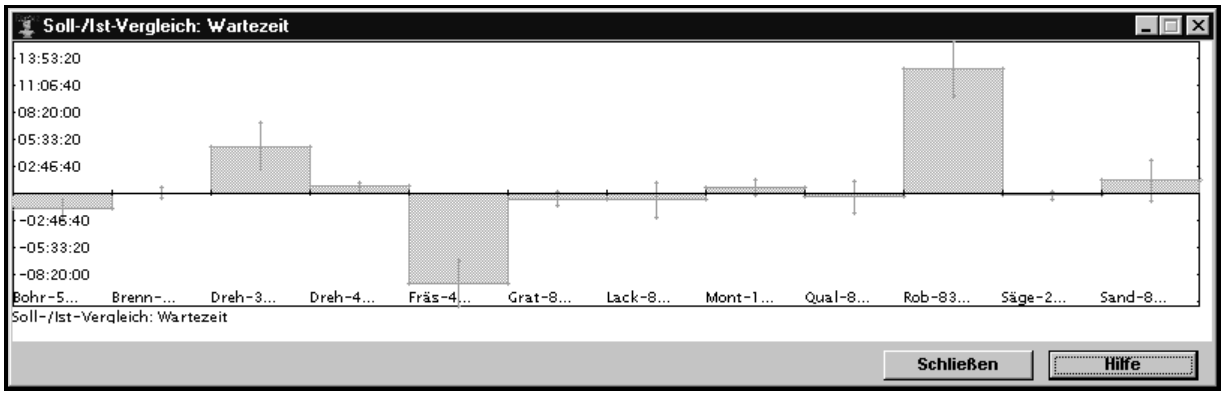


Abbildung 41: Handbuch - Soll-Ist-Vergleich (Diagramm)

Auslastungstabelle

Die Auslastungstabelle gibt eine Übersicht über die Auslastung der Ressourcen während der Simulation. Die Auslastung kann dabei in verschiedener zeitlicher Auflösung und Detailstufen dargestellt werden. Ebenso können alle Positionen, die auf den einzelnen Ressourcen bearbeitet wurden, betrachtet werden. Die Anzeige ist eine WIZARD-Darstellung, die durch die Integrationsfunktionen mit den Simulationsdaten gefüllt wird. Parallel dazu kann die Auslastungstabelle mit den Planungsdaten betrachtet werden, so daß Vergleiche möglich sind.

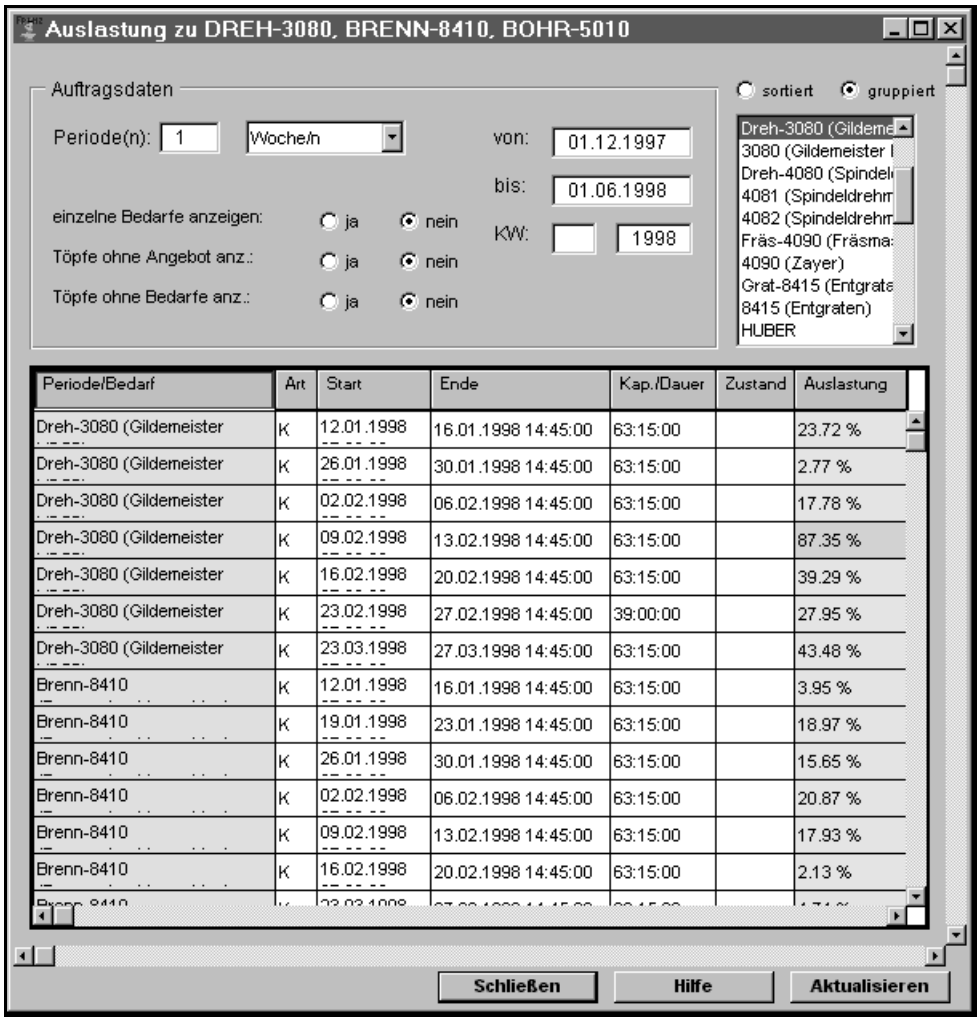


Abbildung 42: Handbuch - Auslastungstabelle

Anhang C: Zufallszahlenerzeugung

Um stochastische Simulationsmodelle simulieren zu können, müssen verschiedene Verteilungsfunktionen bereitgestellt werden. In diesem Abschnitt werden die grundlegenden Algorithmen zur Erzeugung von Zufallszahlen, die nach einer vorgegebenen Verteilungsfunktion *F* verteilt sind, und die in DYBBS implementierten Verteilungsfunktionen vorgestellt.

Eigenschaften von Algorithmen zur Zufallszahlenerzeugung

Algorithmen zur Erzeugung von Zufallszahlen müssen drei Eigenschaften vorweisen können: Exaktheit, Effizienz und Wiederholbarkeit. Unter Exaktheit ist eine möglichst gute Annäherung an die zugrundeliegende Verteilungsfunktion zu verstehen. Die Güte der erzeugten Zufallszahlen läßt sich durch die Anwendung statistischer Tests zeigen. Eine detaillierte Beschreibung der Testverfahren findet sich in [Hoo89] und [Law91]. Die Effizienz betrifft sowohl die Laufzeit- als auch die Speicherplatzkomplexität, da die Zufallsroutinen sehr häufig aufgerufen werden. Unter Wiederholbarkeit ist zu verstehen, daß ein Algorithmus nach einer geeigneten Initialisierung einen Strom von Zufallszahlen reproduzieren kann, was den Vergleich von Experimenten erleichtert.

Erzeugung von (0,1)-gleichverteilten Zufallszahlen

Die Grundlage zur Implementierung von Verteilungsfunktionen bildet die Erzeugung von gleichmäßig über dem Intervall *(0,1)* verteilten Zufallszahlen. Diese werden von den in den folgenden Abschnitten vorgestellten Algorithmen dazu benutzt, um aus der Kombination und Verrechnung von solchen einfachen Zufallszahlen komplexere Verteilungsfunktionen zu erzeugen. Zur computerbasierten Erzeugung von Zufallszahlen wird in der Regel ein algorithmischer Ansatz verwendet. Der einfachste Algorithmus ist die lineare Kongruenzmethode, die eine Zufallszahl rekursiv durch die Formel $Z_i = \left(A \cdot Z_{i-1} + C\right) \bmod M$ mit geeigneten Werten für *A, C* und *M* berechnet. Durch die Angabe eines Startwertes Z_0 liefert dieser Algorithmus reproduzierbare Zufallsströme und es läßt sich für geeignete Parameter beweisen, daß die Wiederholung der Zahlenfolge erst nach *M* Zahlen erfolgt (z.B. für $A=7^5$, $M=2^{31}$, $C=0$, was zusätzlich eine Rechenoperation spart, [Hoo89], S. 249). Für weitere Methoden zur Erzeugung *(0,1)*-verteilter Zufallszahlen muß auf [Hoo89] und [Law91] verwiesen werden. Eine zufällige Variable *X*, deren Verteilungsfunktion gleichmäßig im Intervall *(0,1)* ist, wird im weiteren mit *X~U(0,1)* notiert.

In den meisten Programmiersprachen ist ein solcher Algorithmus bereits integriert, so daß er nicht selbst implementiert werden braucht. In Common-LISP sind dies die Funktionen `random` und `make-random-state`. Die Implementierung der Funktionen ist zwar nicht festgelegt, aber es werden Vorschläge für die Implementierung der Funktionen gemacht ([Ste90], S. 365ff). In DYBBS werden diese Funktionen zur Erzeugung von *U(0,1)*-verteilten Zufallszahlen verwendet.

Algorithmen zur Erzeugung komplexer Verteilungen

Um Zufallszahlen zu erzeugen, die nach einer komplizierteren Verteilungsfunktion verteilt sind, existieren verschiedene Methoden, die je nach Art und Eigenschaft der Verteilungsfunktion eingesetzt werden. Allen Methoden liegt zugrunde, daß sie *U(0,1)*-verteilte Zufallszahlen benötigen und bestimmte Eigenschaften der Verteilungsfunktion ausnutzen:

- **Inversion:** Wenn eine zufällige Variable *X* kontinuierlich ist und nach einer kontinuierlichen und streng monoton steigenden Verteilungsfunktion *F* verteilt ist, dann existiert die Inverse der Verteilungsfunktion F^{-1} mit Definitionsbereich *(0,1)*. Der Algorithmus ist somit wie folgt:

  1. Generiere U~U(0,1)
  2. Berechne $X=F^{-1}(U)$

- **Komposition:** Der Kompositionsansatz kann angewendet werden, wenn die gesuchte Verteilungsfunktion *F* als konvexe Kombination von Verteilungsfunktionen $F_1, F_2, \ldots$ dargestellt werden kann und die Einzelverteilungen leichter zu erzeugen sind als *F*. Für die Verteilungsfunktion

  $F(x) = \sum_{j=1}^{\infty} p_j F_j(x), p_j \geq 0, \sum_{j=1}^{\infty} p_j = 1$ ergibt sich folgender Algorithmus:

  1. Generiere eine zufällige Integerzahl j mit P(J=j)=p_j
  2. Berechne X~F_j

- **Faltung:** Wenn die Verteilungsfunktion *F* der gesuchten zufälligen Variablen *X* die Faltung der Verteilungsfunktionen einer endlichen Anzahl anderer identischer und gleichverteilter zufälliger Variablen (IID) mit Verteilungsfunktion *G* darstellt, so läßt sich der Faltungsansatz anwenden. Ist also $X = Y_1 + \ldots + Y_m$, so ist der Algorithmus wie folgt:

  1. Generiere $Y_1, \ldots, Y_m$ alle nach der Verteilungsfunktion G
  2. Berechne $X = Y_1 + \ldots + Y_m$

- **Acceptance-Rejection:** Kann eine Majorante *t* der Verteilungsdichtefunktion *f* gefunden werden mit $c = \int_{-\infty}^{\infty} t(x)dx < \infty$, so ist $r(x) = \frac{t(x)}{c}$ ebenfalls eine Verteilungsdichtefunktion. Können für eine zufällige Variable *Y* mit Dichtefunktion *r* einfach Zufallszahlen generiert werden, so kann die Acceptance-Rejection-Methode angewendet werden:

  1. Generiere Y nach r
  2. Generiere U~U(0,1), unabhängig von Y
  3. Falls U<=F(Y)/T(Y), dann setze X=Y, ansonsten zurück zu 1.

  Dieser Algorithmus ist effizient, wenn nur wenige Fehlversuche benötigt werden. Dazu muß die Majorante *r* möglichst dicht an *f* liegen, d.h. $E\left[\frac{f(Y)}{r(Y)}\right]$ muß maximal sein. Die Acceptance-Rejection-Methode ist vom Grundprinzip ähnlich zur Monte-Carlo-Simulation zur Bestimmung von Integralen.

Details zu den Algorithmen und Beweise über die Korrektheit der Algorithmen werden in [Hoo89] und [Law91] erläutert. Dort werden auch Anwendungen der Algorithmen auf die häufigsten Verteilungsfunktionen dargestellt. Die in DYBBS implementierten Zufallsgeneratoren wurden nach diesen Beispielen implementiert. In Tabelle 9 ist eine Auflistung der implementierten Verteilungsfunktionen und der benötigten Parameter dargestellt. Für eine genaue Beschreibung der Verteilungsfunktionen und ihrer Eigenschaften muß auf [Hoo89] und [Law91] verwiesen werden.

Name/DyBBS-Klasse	**Beschreibung**	**Parameter**	**Methode**
Bernoulli BERNOULLI	Münzwurf mit Erfolgswahrscheinlichkeit	Erfolgswahrscheinlichkeit *p*	Inversion
Binomial BINOMIAL	n-fache Wiederholung eines Münzwurfs	*n* Wiederholungen mit Einzelwahrscheinlichkeit *p*	Faltung
Binomial approximiert BINOMIALAPPROXIMATION	ungenauer, aber schneller als die obige Variante	*n* Wiederholungen mit Einzelwahrscheinlichkeit *p*	Normalverteilung
Deterministisch DETERMINISTIC	keine Zufälligkeit	Mittelwert *m*	-
Dreieck TRIANGULAR	ohne genaueres Wissen über den Verlauf	Intervallgrenzen *a* und *c*, Spitze *b*	Komposition
Empirisch EMPIRIC	Anpassung an Messwerte	Sprungstellen x_j und Funktionswerte $F(x_j)$	Inversion
Erlang-k ERLANG	k-fache Hintereinanderausführung von exponentiellen Phasen	*k* Phasen mit Rate *l*	Faltung
Exponential EXPONENTIAL	Negativ-exponentiell verteilte Zufallszahlen	Rate *l*	Inversion
Geometrisch GEOMETRIC	Anzahl der Versuche bis zum 1. Erfolg bei Münzwurfexperiment	Erfolgswahrscheinlichkeit *p*	Faltung
Gleichmäßig UNIFORM	gleichwahrscheinliche Zufallszahlen über einem Intervall	Intervallgrenzen *a* und *b*	U(0,1)
Gleichmäßig diskret DISCRETEUNIFORM	gleichwahrscheinliche Zufallszahlen über einem diskreten Wertebereich	Intervallgrenzen *a* und *b*	U(0,1)
Hyperexponentiell HYPEREXPONENTIAL	k-fache Parallelisierung einer Exponentialverteilung	*k* Phasen mit Wahrscheinlichkeit p_j und Rate l_j	Komposition
Normal NORMAL	Gauß'sche Glocke	Mittelwert *m*, Standardabweichung *s*	Accept-Reject
Normal approximiert NORMALAPPROXIMATION	ungenauer, aber schneller als die obige Variante	Mittelwert *m*, Standardabweichung *s*	Faltung
Poisson POISSON	Anzahl der Ankünfte bei exponentiell verteiltem Ankunftsstrom während eines festen Zeitraums	Rate *l*	Faltung
Zusammengesetzt COMPOSITE	Summe von unabhängigen ZV	*k* Verteilungsfunktionen, bei der die Verteilungsfunktion D_j k_j-fach gefaltet wird	Faltung

Tabelle 9: In DyBBS implementierte Verteilungsfunktionen

Anhang D: Schnittstellenbeschreibung

Im folgenden werden die wichtigsten Schnittstellen von DYBBS vorgestellt. Diese Aufstellung umfasst nur die wichtigsten Funktionen der einzelnen Module, die in Abschnitt 6.3 vorgestellt wurden. Ebenso sind dort die Klassen der Simulationsumgebung erläutert. Eine detaillierte, automatisch erstellte Quelltextdokumentation ist auf der CD zu finden.

Statistikmodul (Package :randomnumbers)

`SWE=VERTEILUNG-BESTIMMEN`
`distribution [DISTRIBUTION] &key type [:NUMBER]`
Dialog zum Auswählen einer Verteilungsfunktion. distribution kann eine Instanz oder NIL sein. type kann :number oder :time sein, was für das Ausgabeformat der Parameter verwendet wird.

`CREATE...DISTRIBUTION`
`parameters &key precision [ *PRECISION* ] type [ :NUMBER ]`
Für jede der implementierten Verteilungsfunktionen exististert eine Funktion mit den jeweiligen Parametern der Verteilung. Die implementierten Verteilungsfunktionen sind in Tabelle 9 aufgelistet. type kann wiederum :number oder :time sein. precision gibt eine maximale Genauigkeit in Dezimalstellen an.

`MAKETESTDISTRIBUTION`
`name &optional type [ :NUMBER ]`
erzeugt Beispielverteilung vom Typ name mit Default-Parametern. Die gültigen Namen liefert die Funktion GETALLDISTRIBUTIONNAMES. type ist analog zu oben.

`CREATERANDOMNUMBER`
`self [DISTRIBUTION]`
Erzeugt eine nach 'Distribution' verteilte Zufallszahl

`MEAN`
`self [STATISTICSAMPLE]`
empirischer Mittelwert der Stichprobe
`self [DISTRIBUTION]`
wahrer Mittelwert der Verteilungsfunktion

`VARIANCE`
`self [STATISTICSAMPLE]`
Schätzfunktion der Varianz aufgrund der Stichprobe
`self [DISTRIBUTION]`
wahre Varianz der Verteilungsfunktion

`CREATESAMPLE`
`distribution [DISTRIBUTION] &optional max elementfunction`
erzeugt eine leere Stichprobe mit maximalem Umfang max der Verteilung distribution, elementFunction kann eine Zugriffsfunktion auf den Wert des Elements spezifizieren. Ist die Stichprobe nach keiner Verteilung verteilt, so ist für distribution NIL anzugeben.

Simulationskern (Package :cl-user)

`SWE=SIMULATION-STARTEN`
`&optional simulator [ (GETSIMULATOR) ]`
führt einen Simulationslauf durch. Zuvor müssen die Simulationsoptionen auf sinvolle Werte gesetzt sein.

`SWE=SIMULATIONSOPTIONEN`
`&optional allow-all [ T ] simulator [ (GETSIMULATOR) ]`
Dialog zum Einstellen der Ablaufoptionen. allow-all wird während des Simulationslaufs gelöscht, da hier nicht alle Optionen zur Verfügung stehen.

`SWE=SYSTEM-OPTIONEN`
-
Dialog zum Einstellen der Systemoptionen. Diese betreffen hauptsächlich Farben und Icons.

Wissenserwerb und Modellierung (Package :cl-user)

SWE=NEUE-SIMULATIONSUMGEBUNG

&key minimal [NIL]

Erstellen einer minimalen Simulationsumgebung. minimal=T löscht alle Ereignistypen und Zustände bis auf die vom System benötigten.

SWE=NACHFOLGERGRAPH

&optional elemente

Übersicht über alle Elemente, wenn zu viele (> 30 Elemente auf der obersten Ebene) im Modell vorliegen, erfolgt ein Auswahlmenüs, weil der Nachfolgergraph zu unübersichtlich wird. elemente kann eine beliebige Liste von Simulationselementen enthalten, bei NIL werden alle im Simulationselemente auf der obersten Hierarchiestufe angezeigt.

SWE=ZUSTANDSGRAPH

elemente [LIST]

Öffnet Modellierungsfenster zum Editieren des Zustandsgraphen für mehrere Elemente. Bei mehreren Elementen muss ein Referenzelement bestimmt werden.

element [SYMBOL]

Öffnet Modellierungsfenster zum Editieren des Zustandsgraphen für ein Element.

SWE=KLASSENMODELL-ERSTELLEN

-

Dialog zur Auswahl einer Elementklasse und Modellierung des Zustandsgraphen für diese Klasse. Der Zustandsgraph für die Klasse ist zu Beginn leer.

SWE=REGEL-EDITIEREN

regel

Dialog zum Editieren einer Regel. Alle in der Regel vorhandenen Attribute können im Dialog festgelegt werden. Weitere Funktionen in diesem Zusammenhang sind SWE=REGEL-LOESCHEN, SWE=NEUE-REGEL, SWE=REGEL-KOPIEREN.

SWE=VORBEDINGUNG-EDITIEREN

vorbedingung

Dialog zum Editieren einer Vorbedingung.

SWE=AUSWIRKUNG-EDITIEREN

auswirkung

Dialog zum Editieren einer Auswirkung. Zuerst wird der Typ der Auswirkung bestimmt, daraufhin kommt je nach Typ ein spezieller Dialog zur Eingabe der Attribute.

SWE=RESSOURCE-EDITIEREN

element

Dialog zum Editieren eines Simulationselements. Je nach Typ des Elements stehen verschiedene Felder zur Verfügung (Warteschlange, Bewertungsfunktion, Nachfolger). Es können auch mehrere Elemente gleichzeitig editiert werden, diese müssen jedoch die gleiche Klasse besitzen. Beim Schliessen des Dialogs wird gefragt, welche Attribute für alle Elemente übernommen werden sollen.

SWE=MODELL-UEBERPRUEFEN

&optional simulator [(GETSIMULATOR)]

führt verschiedene Konsistenzüberprüfungen am Modell und der WB durch.Entdeckt werden falsche Parameterverwendungen (nur Anzahl der Parameter), falsche Zustände und Ereignistypen und fehlende Regeln und Auswirkungen.

Ergebnisanalyse (Package :cl-user)

SWE=ERGEBNIS-ANALYSE-GRAPHISCH

&optional deviationanalysis

Stellt Ist-Soll-Vergleich im Diagramm dar. Eine vorhandenes Analyse kann übergeben werden, ansonsten wird eine Auswahl angeboten.

SWE=ERGEBNIS-ANALYSE-IN-TABELLE

&key simulator [(GETSIMULATOR)] analyse title

Gibt Ist-/Soll-Vergleich in einer Tabelle aus. title ist der Fenstertitel, fehlt analyse, so wird ein Auswahldialog geöffnet.

SWE=RESSOURCEN-AUSLASTUNGS-TABELLE

&rest ressourcen

Simulationsauslastung wie in pps=ressourcen-auslastungs-tabelle mit kapazitaetstoepfen, falls mehr als ein Simulationslauf, dann kommt die alte Auslastungsberechnung ueber die Gesamtzeit ohne Toepfe. Eine Liste der Ressourcen kann übergeben werden.

SWE=SIMULATIONSERGEBNISSE-AUSGEBEN-IN-TABELLE

&optional simulator [(GETSIMULATOR)]

Gibt Simulationsergebnisse in einer Tabelle aus. Ergebnisse sind die gleichen wie in swe=simulationsergebnisse-ausgeben, d.h. es wird für jedes Element eine Statistik über die einzelnen Zustände berechnet und präsentiert.

SWE=SIMULATIONSERGEBNISSE-GRAPHISCH

&optional resources simulator [(GETSIMULATOR)]

Stellt Ergebnisstatistik im Diagramm dar. Die Eregbnisse sind dieselben wie oben, nur das der Mittelwert für jeden Zustand sowie die Warteschlangenlänge als Balkendiagramme gezeichnet werden.

Integration in WIZARD (Package :cl-user)

SWE=RESSOURCE-HINZUFUEGEN

&optional show-p [T]

Dialog zum Hinzufügen eines Elements in Simulationsmodell. Das Element muss existieren (z.B. durch Anlegen des Objekts oder Koppeln). show-p=T aktualisiert sofort alle Graphenfenster. Dieses wird beim Laden eines Simulationsmodells ausgeschaltet.

SWE=WIZARD-BEWERTUNG-DURCHFUEHREN

&optional simulator [(GETSIMULATOR)] szenario

führt Bewertung nach WIZARD-Constraints durch. szenario wird noch ignoriert. Die Bewertung erfolgt anhand der Simulationsattribute in den Positionen und Operationen sowie der Belegungsstruktur in den Ressourcen.

SWE=WIZARD-UMPLANUNG-DURCHFUEHREN

operationen &optional simulator [(GETSIMULATOR)] szenario

führt Umplanung nach WIZARD-Constraints durch. Defaultszenario ist das Umplanungsbenutzerprofil. operationen ist die Liste der umzuplanenden Operationen. Alle Operationen, die nicht umgeplant werden dürfen, müssen mit wr=fixieren festgehalten werden.

SWE=WIZARD-UMPLANUNG-DURCHFUEHREN

operationen &optional simulator [(GETSIMULATOR)] szenario

führt Umplanung nach WIZARD-Constraints durch. Defaultszenario ist das Umplanungsbenutzerprofil. operationen ist die Liste der umzuplanenden Operationen. Alle Operationen, die nicht umgeplant werden dürfen, müssen mit wr=fixieren festgehalten werden.

SWE=POSITION-BELEGEN

objekt &key mit-kap-toefen [T]

trägt für eine Position oder Operation alle Simulationsattribute in die Belegungsstruktur ein. Diese sind Dauer, Zeitpunkt und Ressource. Bei mit-kap-toepfen=T werden auch die Kapazitätstöpfe der Ressource aktualisiert. Analog arbietet SWE=POSITION-FREIGEBEN.

Anhang E: Danksagung

> *„Was wir zusammen vortragen, mag von A bis Z falsch sein, aber wir sprechen es mit großem Selbstvertrauen aus. Wir hoffen, daß unsere Ideen, auch wenn sie falsch sein sollten, Ihnen neue Orte erschließen, die einen Besuch wert sind. Wir glauben, daß wir dort, wo wir unrecht haben, auf konstruktive Weise unrecht haben, und daß unsere Irrtümer aufschlußreicher sind als die richtigen Ansichten der orthodoxen Theorie." (Jack Cohen & Ian Stewart „Chaos und Anti-Chaos", S. 13)*

Diese Arbeit wäre ohne die Mithilfe, Ideen, Ratschläge, An- und Abwesenheit, Aufmunterungen, Spott, Nebenbemerkungen, Ablenkungen, Spiele-Sessions und Kneipenabende der folgenden Personen nie zu dem geworden, was sie jetzt ist. Also gilt mein Dank (in alphabetischer Reihenfolge, damit keiner daraus irgendwelche Schlüsse ziehen kann):

Stefan Bamberger, Christian Betz, Petra Braun, Ciske Busch, Dogbert, Helge Hargesheimer, Rainer Herrler, Jens Jordan, Franzi Klügl, Ioannis Iglezakis, Stefan H. Landvogt, Christoph Oechslein, Stefan Raspl, Michael Ratz, Tina Reinhardt, Markus „Topsi" Tolksdorf, Michael Wolber

Michael Wolber danke ich zusätzlich für seine Fähigkeiten, völlig mysteriöse Fehler in seinen Sourcen beheben zu können. Christoph Oechslein und Christian Betz sei dafür gedankt, daß sie die körperlichen und seelischen Belastungen des Korrekturlesens auf sich genommen haben. Speziell für euch und meinen Betreuer noch einer: Diese Satz drei Fehler.

Christian Hestermann zeigte bei der Betreuung dieser Arbeit viel Geduld, v.a. was mein Verständnis von Rechtschreibung und die Erklärung von mißverständlichen Beispielen anging. Für die vielen anregenden und sinnvollen Diskussionen bin ich ihm im Besonderen dankbar. Außerdem bestand er darauf, in der Danksagung erwähnt zu werden, weil das Einfluß auf die Bewertung der Arbeit haben könnte.

Ohne die folgenden Sponsoren wäre das ganze Werk natürlich nie in Angriff genommen worden, weshalb ich mich abschliessend bei ihnen bedanken möchte:

Mami + Papi

Anhang F: CDROM

Für Hinweise zur Installation lesen Sie die Datei `'Readme.txt'` im Hauptverzeichnis der CD.